AUTOMATIC STROKE LESION SEGMENTATION WITH LIMITED MODALITIES

by

Craig W. Fraser

B. S. Electrical Engineering, University of Pittsburgh, 2012

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2016

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Craig W. Fraser

It was defended on

November 29, 2016

and approved by

Steven P. Jacobs, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Murat Akcakaya, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

Ching-Chung Li, Ph.D., Professor, Department of Electrical and Computer Engineering

Thesis Advisor: Steven P. Jacobs, Ph.D., Assistant Professor, Department of Electrical and Computer Engineering

AUTOMATIC STROKE LESION SEGMENTATION WITH LIMITED MODALITIES

Craig W. Fraser, M.S.

University of Pittsburgh, 2016

MRI Brain image segmentation is a valuable tool in the diagnosis and treatment of many different types of brain damage. There is a strong push for development of computerized segmentation algorithms that can automate this process because segmentation by hand requires a great deal of effort by a highly skilled professional. While hand segmentation is currently considered the gold standard, it is not without flaws; for example, segmentation by two different people can provide slightly different results, and segmentation by hand is labor intensive. Due to these flaws, It is desirable to make this process more consistent and more efficient through computer automation.

This project investigates four promising approaches for the automatic segmentation of brain MRIs containing stroke lesions found in recent literature. Two of these algorithms are designed to use multiple modalities of the same patient during segmentation, while the other two are designed to handle one specific modality. The robustness of each to limited, or different, image sequences than they were originally designed for will be tested by applying each to two datasets that contain 24 and 36 patients with chronic stroke lesions.

These tests concluded that performance for the multi modal algorithms does tend to decrease as input modalities are removed, however it also revealed that FLAIR imaging in particular seems to be especially valuable for segmenting stroke lesions. In both multi-modal algorithms while there was an overall drop in Dice scores, any testing that included FLAIR images performed significantly better than any other tests. The single channel algorithms had difficulty segmenting any modalities different from the specific one that they were designed for, and were generally unable to detect very small lesions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.0 INTRODUCTION

In this document we present several state of the art tecniques for automatic stroke lesion segmentation in brain MRI, and test the efficacy of each using images with both chronic and acute stroke damage. The acute stroke data were taken from the ISLES 2015 challege, and the chronic data were supplied by the University of Pittsburgh Learning Research and Development Center (LRDC).

There are three challenges when applying these methods. First, most of the algorithms discussed were designed to take into account up to four different imaging types, here we apply limited data to measure performance degradation for each approach. Second, most of these algorithms have been designed or trained strictly for acute stroke lesions and may not be generalizable to chronic lesion, as each appears significantly different in an MRI. Finally, for each learning algorithm, models were trained with images from one imaging center then tested using data from a different center, which typically results in some loss of performance.

1.1 PROBLEM STATEMENT

Many automatic lesion segmentation algorithms are designed using several different MRI modalities, or a specific modality, which will frequently not always be available in practical settings. We test several approaches with limited datasets, or different modalities, to see if they are generalizable.

1.2 SCOPE AND OBJECTIVES

We test four algorithms in recent literature for the automatic segmentation of brain MRI containing stroke lesions. The algorithms proposed in [1] and [2] are designed to take advantage of four modalities, and the algorithms proposed in [3] and [4] are designed for a particular modality. Each algorithm was tested using chronic and acute stroke data with three main objectives:

1. Compare performance between multi and single modal algorithms to determine if one is generally superior to the other
2. For multi-modal algorithms, compare performance as fewer modalities are provided to determine how detrimental removing modalities is, and if certain modalities provide more useful information than others

1.3 ORGANIZATION OF THESIS

This document is divided into six chapters, with further detail about each below.

Chapter 2 provides a background about automated lesion segmentation of medical imaging. This includes a brief justification about why fully-automatic lesion segmentation is desirable, as well as a discussion of modern challenges persuing automatic lesion segmentation.

Chapter 3 reviews modern literature representing state of the art approaches for automatic lesion segmentation in brain MRI. Four main papers are discussed which describe each of the algorithms that are tested in this document. A basic explanation of each algorithm is provided, and the results are also discussed here. Additionally a paper describing the most common useful analysis metrics for medical imaging problems is discussed.

Chapter 4 discusses technical details about each of the four algorithms that were tested in greater depth, and how the tests were performed. It also provides a justification for any pre-processing that was performed on the input data prior to segmentation, and describes the analysis metrics selected.

Chapter 5 presents the quantitative results of testing with each algorithm. For multi-modal algorithms, resulting segmentations from all possible input combinations are provided.

Chapter 6 summarizes all of the findings from this thesis, and compares the performance levels for each algorithm acheived here with those published in prior works. Where applicable it discusses possible sources of discrepancy between sources. Finally it provides reccomendations for possible further research related to automatic lesion segmentation of MRI containing stroke lesions.

2.0 BACKGROUND

Brain image segmentation is a very useful tool for computer aided diagnosis of various types of brain damage. For patients with a tumor, stroke, traumatic brain injury, or Multiple Sclerosis information about the size and location of affected regions, and how they change over time can be useful for tracking treatment progress. Researchers are interested in segmentation from both clinical (evaluating efficacy of treatement) and research (learning about brain function) perspectives. Many different research teams have developed new approaches to this problem with varying levels of success and generalizability. It is a very difficult problem, because damaged tissue may produce image intensities that are similar to some types of healthy tissue, and each kind of brain damage presents differently in MRIs. As a result certain algorithms that are successful with one type may not be as useful for another.

Brain segmentation faces a number of challenges. Currently the standard is manual segmentation by a highly-skilled professional. While extremely useful, manual segmentations can differ significantly between individual professionals, and the results may not always be consistent, even between segmentations of the same image set by the same person. Manual segmentation is highly labor intensive, requiring the full focus of a skilled professional for a significant amount of time.

Due to those drawbacks, computerized automatic segmentations are highly desirable. Not only do they free up time that would be spent manually segmenting, but they may provide a more consistent result between patients and between subsequent segmentations. Currently, many automatic segmentation approaches are very resource intensive, but this will become less of an issue as algorithms are streamlined, and computer technology improves.

There are presently many promising approaches to this problem in the engineering and medical literature, mostly focused on brain tumors. In [5] Liu et al. provide a survey of many state of the art methods for automatically segmenting MRI images containing brain tumors. They classify

each algorithm as one of three types: Conventional, Classification and Clustering, and Deformable Model methods. Conventional methods are characterized by the use of typical image processing techniques, such as threshold and region-based algorithms. Classification and Clustering methods are machine learning techniques where either a labeled or an unlabeled data set is used to train the algorithm. In a supervised learning method, a labeled dataset is used to train the algorithm, the end goal being a generalized result that can be applied to an unlabeled dataset successfully. Unsupervised learning does not use a pre-labeled dataset, and hopes to reveal hidden relationships between samples using only one dataset. Deformable models attempt to identify the boundaries of a structure in a 3-D MRI, then use those boundaries to construct a model of those structures. These methods support the use of a priori knowledge about those structures, and are able to handle variability that may occur in anatomical structures due to time, or across several patients.

[6] summarizes the results of the 2012 and 2013 Multimodal Brain Tumor Segmentation Challenge (BRATS) and draws some conclusions about the efficacy of current algorithms. BRATS takes several algorithms and applies them to both a real clinical dataset containing brain tumors, and a realistic simulation of tumor images. Each research group is given some subset of the data and allowed to optimize their algorithm using that data, then the now-optimized algorithims are applied to the rest of the data. BRATS calculates the accuracy of each algorithm by comparing their results with a consesus of segmentations by human experts and calculating the Dice score, which is a statistic of similarity between two sets that when applied to image segmentations measures spatial overlap between two segmentations of the same image. From the results of BRATS 2012 and 2013 Menze et al. notice that the best performing algorithms tend to use discriminative learning approaches, and that the worst performing algorithms tended to rely on basic images features. They suggest that the algorithms that perform poorly suffer from high rates of false positives and may benefit from spatial regularization in post-processing that would remove those false positives, increasing their overall Dice score. Finally they notice that fusing segmentations from multiple approaches always results in a superior segmentation than the results from any single algorithm. This is a well known concept from machine learning which is used to reduce error when there is a high rate of variability betwen different predictors.

In 2015 a similar competition known as as the Ischemic Stroke Lesion Segmentation Challenge (ISLES) began that tasks researchers in a similar way to BRATS, but uses MRI data containing stroke lesions instead of brain tumors. Organized similarly to BRATS, ISLES provides each team with a training and testing dataset consisting of four different MRI modalities: Spin-Lattice relaxation time (T1), Spin-Spin relaxation time (T2), Diffusion Weighted MRI (DWI), and Fluid-Attenuated Inversion Recovery (FLAIR). Until ISLES began in 2015 little medical image segmentation research was focused on stroke lesions, most were trying to detect brain tumors. In the last 2 years more researchers have begun to look at stroke lesions, but the volume of work still lags behind that of tumors. Thanks to yearly competitions such as BRATS and ISLES, this technology improves every year, and fully automated reliable segmentations may become a reality in the near future. This report focuses on four promising algorithms detailed in [1, 7, 3, 4]

3.0 LITERATURE REVIEW

The state of the art in medical image segmentation has been rapidly advancing in recent years partially due to the efforts of the yearly BRATS competetions since 2012, and the ISLES competitions since 2015. While approaches to solving the problem of automated segmentation have been proposed for years, these challenges bring the most promising algorithms together and test them all against a common dataset. Where previously they would only be tested against private datasets, application of a wide variety of approaches to a single common dataset allows researchers to directly compare their performance in a way never before possible. The DeepMedic algorithm was not entered in BRATS 2015, but Glocker et al. assert that their acheived average Dice score of 84.7 on the BRATS data would have ranked highly in [1], and that same algorithm won the ISLES 2015 challenge. Several other top-performing techniques in ISLES 2015 used machine learning and clustering techniques similar to those discussed in [5].

3.1 GENERATIVE MODEL WITH BIOLOGICAL CONSTRAINTS

Menze et al. have been developing algorithms for automatic segmentation of brain tumors for several years. In [2] they propose a generative probabilistic model for segmentation of tumors in multimodal MRI data. They claim that their algorithm is applicable to any set of multimodal data, and allows for different tumor boundaries in each modality, or channel, of the data, meaning that their model should be able to handle a lesion that appears differently in each channel.

The basic model takes advantage of minimal spatial context, as the tumor state at a given voxel is assumed to be independent of the tumor state in other voxels. Menze et al develop three extensions to their model and test the results of each. First, they extend their spatial tumor prior

(atlas) to include a Markov Random Field prior, which takes in to account the six nearest neighbors of each voxel and defines how similar their tumor states tend to be. Next, they allow for multiple tissue classes within their tumor class to represent tumor substructures, such as edema. Finally, they relax the independence between data channels by switching their data likelihood equation to use multivariate Gaussians.

Menze et al. have notable success with their algorithm, and when compared to two alternate EM segmentation methods theirs increases the Dice score by 0.1-0.2 for all modalities tested. They continue their research in [7] by extending the generative model they developed in [2]. They extend and further test their generative model in 3 ways. First, they propose a new generative model that shares information about lesion location between the different imaging modalities. Next, they test the generalizability of their algorithm by applying it directly to stroke images. Finally, they add a discriminative method that allows the algorithm to generate useful tissue labels at each point, as opposed to just identification of hyper- and hypo-intensities in eash tissue type, as the basic generative model produces.

The generative model extensions are based on prior biological knowledge about tumor growth. They added two conditional dependencies on tumor occurrence where their previous model had assumed independence. The calculation of label vectors is limited to only biologically plausible combinations by assuming the conditional dependence that areas which contain Cerebro-spinal Fluid (CSF) in one channel do not contain tumor in another channel. Menze et al. also account for expected appearances in each MRI modality they used, T1, T1 with contast material (T1c), T2, and FLAIR. They require that edema visible in T2 images will always coincide with edema in FLAIR images and that lesions visible in T1 and T1c are always contained within lesions visible in T2 and FLAIR. Additionally, the EM portion of the algorithm now enforces that tumor must present as a hyper or hypo intensity (depending on the image modality) compared to the current average of the white matter segmentation. If the intensity of a suspected tumor voxel does not satisfy this requirement, its probability is set to 0. Finally, they also add in the Markov Random Field extension that was discussed and tested in [2], which enforces spatial regularity to reduce the false positive rate.

Even with these extensions the generative model still only takes limited local information into account. Menze et al. attempt to improve this by adding two different discriminative methods after the generative model is applied to data. The first determines whether a given region is an actual tumor or a typical artifact, effectively filtering out false positives, and the second allows the algorithm to interpret regions that are not necessarily hyper- or hypo-intense segments of each channel, allowing for segmentation into more specific regions such as necrosis or non-enhancing core. The addition of these extensions shows a marked improvement over the basic generative model presented in [2]. The extensions to the generative model make it more interesting because they add some dependence on multimodal data. As of this writing they have not tested this approach against limited datasets that only have a few of the channels available.

After all of the additional development in [7] and testing on their brain tumor data, Menze et al. also test their algorithm on stroke lesions with great success. The only edits made to adjust their code for the stroke images is to relax the requirement that all lesion voxels detected in the T1c images must be fully contained within the lesion volume detected in the T2 and FLAIR channels. When applied to the stroke dataset with T1, T1c, T2, and FLAIR images their algorithm acheives an average Dice score of 0.78.

3.2 SEGMENTATION USING CONVOLUTIONAL NEURAL NETWORKS

A useful approach for 3-dimensional image segmentation is that of Convolutional Neural Networks (CNNs), a state of the art machine learning technique. Glocker et al. developed a segmentation program named DeepMedic, which uses a 3-D CNN and a fully-connected Conditional Random Field (CRF), marking the first time such a CRF was applied to medical data. Their approach was very successful, achieving high ranking results in BRATS 2015, and winning ISLES 2015.

Glocker et al. extend existing 2D CNNs already found to be useful for image segmentation in two major ways. First, they propose a novel training scheme for the CNN utilizing a technique known as deep training; however, they find the full deep learning technique proposed cannot be implemented successfully due to memory restrictions. Instead, they propose a hybrid scheme. Convolutions with 3D kernels are far more computationally intense than convolutions with 2D

kernels, so they reduce the kernel size from the typical 5^3 to a smaller 3^3, and reduce the negative effects of this smaller kernel size by increasing the number of layers in the network proportionally. Second, their algorithm simultaneously operates on the main image, as well as a down-sampled version of the image, allowing their approach to incorporate more contextual information into the segmentation process. Each pathway is able to extract features from the images independently, meaning that the full resolution pathway is able to learn fine detail, while the down-sampled pathway learns larger bulk structures. As the final post-processing step, the CRF is able to remove many of the false positive structures that the CNN identifies as lesion by itself, improving the overall accuracy of the system.

In [1], Glocker et al. test against BRATS and ISLES data, which contain several modalities. While the algorithm performs very well in each case, there is no mention of segmenting limited modality data. Since the algorithm takes all modalities into account during the learning process it may be interesting to see if a model trained with several modalities is as successful segmenting data where only one, or a few, of those modalities are present, or if a model trained with only a single modality is still as effective.

3.3 DISCRIMINATING SEGMENTS BY HISTOGRAM MAXIMA

In [3] Nooshin et al. propose the novel Histogram-Based Gravitational Optimization Algorithm (HGOA). This algorithm approaches the problem of limited-modality data by using only a single modality, Diffusion Weighted T1. The algorithm is applied to both a simulated tumor dataset, and a real stroke dataset and claims very high accuracy in each case. This algorithm is based on the assumption that the local maxima of the image histogram correspond to each structure in the brain, meaning that the number and intensity of local maxima relate to the number of segments, and their aveage intensity.

The algorithm in [3] effectively consists of two different parts: the histogram-based brain segmentation, and the n-dimensional gravitational optimization algorithm. The basic operation flow is to select the desired number of brain segments (i.e. tissue types), and the number of iterations, then run the n-dimensional gravitational optimization algorithm against the objective function, which is the result of a histogram-based brain segmentation algorithm until the number of desired segments is achieved. The histogram-based segmentation algorithm is a seven step process shown below:

1. Generate the intensity histogram of the image
2. Apply a weighted averaging technique to the histogram
3. Extract the local maxima
4. Convolve a rectangular window with the local maxima
5. Obtain the lower and upper intensity bounds that will define each segment according to a threshold
6. Connect the borders of each segment proportionally
7. Assign an intensity value to each segment

Nooshin et al's. algorithm appears very promising, but is only tested against Diffusion Weighted T1 images. It would be interesting to test this algorithm with other image modalities, to see if it is overly optimized for the particular modality they used.

3.4 SEGMENTATION BY NEIGHBORHOOD DATA ANALYSIS

Pustina et al. propose a novel algorithm known as Lesion Identification with Neighborhood Data Analysis or LINDA in [4]. LINDA was trained using a set of T1 MRI images containing left hemispheric strokes from 60 patients that had been manually segmented by experts. Using these images and their manual segmentations as ground truths, Pustina et al. trained a model by performing an iterative process starting with downsampled low resolution images, and progressing to the full resolution original images. At each resolution step, the algorithm constructs a matrix from all the patient data where each row corresponds to an individual voxel, and columns correspond to information about its neighboring voxels by calulating 12 key features discussed in [4]. This matrix is

used to train a random forest model, which is then applied to those same images to generate a set of posterior probability maps. The output of the lower resolution step is used as a set of additional features for the next higher resolution step, and this process repeats up to the highest resolution.

Once a fully trained model is complete it can be used to segment new patients using a very similar workflow, with the addition of multiple registration and prediction steps. To segment a new image, it is first registered to a template, then all of the features from [4] are calculated, and used with the trained random forest model to predict the lesion locations, using the same low to high resolution process. After the final prediction is made it is used to improve the original registration, at which point the process is repeated three times. The fully trained model created during the research discussed in [4] and scripts for segmentation have been made freely available to the public.

Once LINDA was trained and able to segment their own data Pustina et al. tested it against an alternative algorithm known as ALI discussed in [8] which depends on an outlier detection approach. ALI produced a lower DICE score, and a higher failure rate against the same dataset vs LINDA. Pustina et al. also applied LINDA to a complimentary dataset from another university. They expected LINDAs success to fall significantly, but they find that LINDAs Dice scores in this case only fall by about 0.02. They beleive that this means LINDA models trained at one lab can be applied to data from another lab very successfully. LINDA has never been tested against the ISLES data, so one cannot make a direct comparison between it and some of the earlier algorithms discussed. Additionally, since LINDA was only trained with T1 imaging it may not be as successful when tested with T2 or FLAIR data, and against algorithms developed to use multimodal data.

3.5 METRICS FOR EVALUATING SEGMENTATIONS

Taha and Hanbury discuss a number of common metrics used for analyzing medical image segmentations in [9]. They selected the 20 most common metrics used in current literature, and describe their calculation, propose efficient implementations, and discuss various lesion qualities that would make particular metrics unreliable, or especially useful.

Taha and Hanbury break the 20 metrics down into six families, spatial overlap based, volume based, pair counting based, information theoretic, probabalistic, and spatial distance based measurements. Spatial overlap based metrics are all derived from the four basic cardinalities that make up the confusion matrix, true positive (P_t), true negative (N_t), false positive (P_f), and false negative (N_f). The most prevalent of the overlap based metrics is the Dice-Sørensen coefficient or Dice score that is frequently used in various medical image segmentation tasks. It measures the overlap between a segmentation, and ground truth as the ratio of true positive detections to the sum of true positive, and all errors in the segmentation, calculated using Equation 3.1 where P_s and P_{GT} are the positive voxels in the segmentation and ground truth respectively.

$$DSC = \frac{2P_t}{P_s + P_{GT}} \tag{3.1}$$

Two other common metrics are the true positive and true negative rate, or sensitivity and specificity, which measure the ratio of true positives to true positives and false negative or true negatives to true negatives and false positives respectively, given as Equations 3.2 and 3.3.

$$SEN = \frac{P_t}{P_t + N_f} \tag{3.2}$$

$$SPC = \frac{N_t}{N_t + P_f} \tag{3.3}$$

Precision is another less common metric which is defined by the ratio of true positives to all positive detections. Presicion is not frequently used directly in image segmentation tasks, but is used to calculate the Dice score, and can be calculated directly using Equation 3.4. The final overlap based metric is the Global Consistency Error, defined as the average error for all voxels. Calculation of GCE is more complicated than most of the other metrics and is shown as Equation 3.5.

$$PRC = \frac{P_t}{P_t + P_f} \tag{3.4}$$

$$GCE = \frac{1}{n} \min \left\{ \frac{N_f(N_f + 2P_t)}{P_t + N_f} + \frac{P_f(P_f + 2N_t)}{N_t + N_f}, \frac{P_f(P_f + 2P_t)}{P_t + P_f} + \frac{N_f(N_f + 2N_t)}{N_t + N_f} \right\} \tag{3.5}$$

There is only one volume based metric, the volumetric similarity. Volumetric similarity between two volumes is defined as one minus their volumetric distance, or the ratio of absolute volume difference between them to the total volume. Volumetric similarity does not depend on overlap between volumes, and could have a very high value when there is no overlap if the segmentation was correct but translated due to some error.

There are two pair counting based metrics, the Rand Index (RI), and Adjusted Rand Index (ARI). Both are constructed of four cardinalities, a, b, c, and d. For two segmentations S_1 and S_2 with voxels s_i and s_j, a defines the set where both s_i and s_j are put in the same partition of both segmentations, b is when s_i and s_j are placed in the same partition of S_1, but different partitions of S_2. c is the set where they are in the same partition of S_2, but different partitions of S_1, and d is where s_i and s_j are placed in different partitions in both segmentations.

Given those pairwise cardinalities, RI measures similarity between clusterings and is computed using Equation 3.6. ARI is then the RI when corrected for the random chance of similarity between the two groupings, calculated using Equation 3.7.

$$RI = \frac{a+b}{a+b+c+d} \tag{3.6}$$

$$ARI = \frac{2(ad-bc)}{c^2+b^2+2ad+(a+d)(b+c)} \tag{3.7}$$

Two typical metrics come from information theory, the mutual information, and variation of information. Mutual information measures how much uncertainty in one variable is reduced if the other is known. For image segmentation tasks the mutual information is calculated between segments (lesion or background) instead of individual voxels, with the probability of each segment defined using the four basic cardinalities of the overlap based methods. Variation of information measures the distance between two sets based on their mutual information, or the amount of information gained when switching from one to the other.

The Inter-Class Correlation (ICC), Probabalistic Difference (PBD), Cohen's Kappa coefficient (KAP), and the Receiver Operating Characteristic (ROC) curve are the four probabalistic metrics discussed in [9]. ICC measures consistency between two segmentations, and is typically used in medical imaging problems to compare the segmentations produced by two different manual

tracers. ICC is calculated using Equation 3.8 where MS_b and MS_w are defined as the mean squares distance between segmentations and the mean squares distance within the segmentation, shown in Equations 3.9 and 3.10 respectively.

$$ICC = \frac{MS_b - MS_w}{MS_b + (k-1)MS_w} \tag{3.8}$$

$$MS_b = \frac{2}{n-1}\sum_x (m(x) - \mu)^2 \tag{3.9}$$

$$MS_w = \frac{1}{n}\sum_x (f_g(x) - m(x))^2 + (f_t(x) - m(x))^2 \tag{3.10}$$

PBD is the distance between two segmentations' probability distributions. When applied on the generated and true segmentations it is calculated using Equation 3.11 where f_g and f_t are the voxel intensities of the generated and "true" segmentation respectively.

$$PBD(S_g, S_t) = \frac{\sum_x |f_g(x) - f_t(x)|}{2\sum_x f_g(x) f_t(x)} \tag{3.11}$$

KAP is the agreement between two segments when corrected for the random chance of agreement, similarly to ICC it is frequently used to compare subsequent manual segmentations. The calculation for KAP is shown in Equation 3.12 where P_a is the agreement between the two segments, and P_c is the probability of random agreement.

$$KAP = \frac{P_a - P_c}{1 - P_c} \tag{3.12}$$

The ROC curve is traditionally generated by making many measurements, and plotting the true positive rate against the false positive rate (FPR), where the area under the ROC curve is interpreted as the probability that the receiver will rank a random positive sample higher than a random negative sample. However, the area under the curve (AUC) for a single measurement is also defined using Equation 3.13, and can be applied for a single measurement of a segmentation compared to the ground truth where FNR denotes the false negative rate.

$$AUC = 1 - \frac{FPR - FNR}{2} \tag{3.13}$$

The final group discussed in [9] are the spatial distance based metrics. These measurments account for the spatial position of voxels in the same segment, and offer a convenient way to determine if the countour of a segmented lesion is accurately defined. Taha and Hanbury describe three common spatial distance metrics used in medical imaging literature, Hausdorff Distance, Average Distance, and Mahalanobis Distance.

The Hausdorff Distance is defined as Equation 3.14, where $h(A, B)$ is the Directed Hausdorff Distance calculated using Equation 3.15. The Hausdorff Distance when applied to image segmentation measures the maximum distance between edges of the segmented lesion, and the true lesion. It is very sensitive to outliers, so it becomes very useful to medical segmentation problems because it will give a measure of how much the edge contour of the segmentation differs from the truth, and can be used to balance the inflation of the Dice score for very large lesions. Since Hausdorff Distance is very sensitive to outliers, which are very common in medical images due to noise, [9] proposes using an average Hausdorff distance instead, referred to as the average distance. Average distance is calculated using the same basic equation as the Hausdorff Distance, Equation 3.14, however in this case the directed Hausdorff distance $h(A, B)$ is instead defined using 3.16.

$$HD(A, B) = \max(h(A, B), h(B, A)) \tag{3.14}$$

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\| \tag{3.15}$$

$$h(A, B) = \frac{1}{N} \sum_{a \in A} \min_{b \in B} \|a - b\| \tag{3.16}$$

Mahalanobis Distance measures the distance between a set of points, and an expected probability distribution. For medical image segmentation it is calculated using the set of points that make up the segmentation result, and the ground truth segmentation is used as the target probability distribution. Equation 3.17 where μ_x and μ_y are the means of the generated and ground truth segmentations respectively and S is the common covariance matrix between them as defined in Equation 3.18.

$$MD(X, Y) = \sqrt{(\mu_x - \mu_y)^T S^{-1} (\mu_x - \mu_y)} \tag{3.17}$$

$$S = \frac{n_x S_x + n_y S_y}{n_x + n_y} \tag{3.18}$$

The yearly ISLES competitions also use one additional metric not covered in [9], Average Symmetric Surface Distance (ASSD). It computes the average of the minimum Euclidian distance between the contours of the generated segmentation, and the manual tracing. It can be calculated using Equation 3.19 where S_A is the automatic segmentation generated and S_M is the manual segmentation used as the ground truth[10].

$$ASSD = \frac{1}{|S_A| + |S_M|} \left(\sum_{s_a \in S_A} d(s_a, S_M) + \sum_{s_b \in S_M} d(s_m, S_A) \right) \tag{3.19}$$

4.0 METHODS

This chapter describes the four algorithms used to segment MRI data containing stroke lesions, the experiments performed with each, the MRI data that were used, and the analysis plan for the final segmentation data. The goal is to analyze the practical efficacy of each algorithm for segmenting chronic stroke lesions, and to measure the decline in performance as fewer input modalities are given. Results from each algorithm are presented in Chapter 5.

4.1 INPUT DATA

The input data currently available for these experiments consists of MRI brain volumes from 28 patients, each of whom suffered from a stroke. Currently, all 28 patients have T1 weighted imaging available, 9 also have T2 weighted imaging, and 23 have another type of T2 weighted imaging known as FLAIR which theoretically cancels out the signal from CSF. All patient imaging was provided as original high-resolution 3-D image volumes saved as multiple Digital Imaging and Communications in Medicine (DICOM) file series, as well as skull-stripped images, and manually segmented images to serve as ground truths in NifTI format by the University of Pittsburgh Learning Research and Development Center (LRDC). The full breakdown of patients and which imaging modalities are available for each is shown in Table 1.

Additionally, since most algorithms were designed for use with acute stroke images, the set of 36 sub-acute ischemic stroke lesion segmentation (SISS) testing images from the ISLES 2015 challenge were also used to test DeepMedic and the Generative Model.

Table 1: Data breakdown per patient

Patient	Available Modalities	Patient	Available Modalities
Censa 201	T1, FLAIR	Censa 219	T1, FLAIR
Censa 202	T1, T2	Censa 220	T1, FLAIR
Censa 203	T1, T2	Censa 221	T1, T2, FLAIR
Censa 204	T1, T2, FLAIR	Censa 222	T1, FLAIR
Censa 205	T1, FLAIR	Censa 223	T1, FLAIR
Censa 206	T1, T2	Censa 224	T1, FLAIR
Censa 207	T1, FLAIR	Censa 225	T1, FLAIR
Censa 208	T1, T2	Censa 227	T1, FLAIR
Censa 210	T1, FLAIR	Censa 228	T1, FLAIR
Censa 212	T1, FLAIR	Censa 229	T1, FLAIR
Censa 214	T1, T2, FLAIR	Censa 303	T1, FLAIR
Censa 215	T1, T2	Censa 304	T1, FLAIR
Censa 216	T1, FLAIR	Censa 306	T1, FLAIR
Censa 217	T1, T2, FLAIR	Censa 307	T1, FLAIR

4.2 DEEPMEDIC

4.2.1 THEORY

In general, CNNs attempt to produce image segmentations by taking the local neighborhood of each voxel into account and processing it with several layers consisting of sequential filters. Each layer consists of C_l feature maps, which are groups of neurons that detect some particular pattern, or feature. Each feature map generates an output image, $\mathbf{y}_l^m$, according to Equation 4.1 where $\mathbf{k}_l^{m,n}$ is a kernel of learned weights, b_l^m is a learned bias, and f is a non-linearity

$$\mathbf{y}_l^m = f\left(\sum_{n=1}^{C_{l-1}} \mathbf{k}_l^{m,n} * \mathbf{y}_{l-1}^n + b_l^m\right) \tag{4.1}$$

At the very beginning of the network, the input to the first layer, $\mathbf{y}_0^n$, is simply the full image after any pre-processing and the input to each subsequent layer is the output from the previous layer. Due to the sequential nature of the network, each higher layer is combining all of the lower layer features, and able to detect more complex patterns.

The final layer is known as a classification layer, because each neuron corresponds to one

of the possible segmentation classes. The output of the classification layer is processed with the position-wise softmax function expressed in Equation 4.2, which produces a predicted posterior for each class. In Equation 4.2 $\mathbf{y}_L^c(\mathbf{x})$ is the activation of the classification layer's c^{th} feature map, at (x, y, z) position $\mathbf{x}$.

$$p_c(\mathbf{x}) = \frac{\exp(\mathbf{y}_L^c(\mathbf{x})}{\sum_{c=1}^{C_L} \exp(\mathbf{y}_L^c(\mathbf{x}))} \tag{4.2}$$

For final post-processing of the segmentations produced by the CNN, Glocker et al. use the fully connected CRF designed in [11] by extending it to three dimensions. The CRF they propose attempts to reduce the false positive rate by removing small isolated regions of the lesion map that can result from noise in the input, or local minima during the training session. For an input image $\mathbf{I}$ and segmentation z the Gibbs energy is calculated using Equation 4.3, where $P(z_i|\mathbf{I})$ is the output of the CNN at voxel i.

$$E(\mathbf{z}) = \sum_i -\log P(z_i|\mathbf{I}) + \sum_{ij, i \neq j} \mu(z_i, z_j) k(\mathbf{f}_i, \mathbf{f}_j) \tag{4.3}$$

In Equation 4.3 the function $k(\mathbf{f}_i, \mathbf{f}_j)$ is a weighted linear combination of the two Gaussian kernels that Glocker et al.'s CRF uses to enforce spatial relationships between voxels, given as Equation 4.4, where w_m is the weight of the m^{th} kernel. The smoothness kernel defined by Equation 4.5 depends upon a diagonal covariance matrix $\sigma_{\alpha,d}$ which defines the size of the neighborhood in which homogeneous voxel labels should be encouraged. The appearance kernel defined by Equation 4.6 depends on the parameter $\sigma_{\gamma,c}$ which defines how strongly to enforce a homogenous appearance between voxels in the area defined by $\sigma_{\beta,d}$ that have the same label.

$$k(\mathbf{f}_i, \mathbf{f}_j) = w_1 k_1(\mathbf{f}_i, \mathbf{f}_j) + w_2 k_2(\mathbf{f}_i, \mathbf{f}_j) \tag{4.4}$$

$$k_1(\mathbf{f}_i, \mathbf{f}_j) = \exp\left(- \sum_{d=(x,y,z)} \frac{|p_{i,d} - p_{j,d}|^2}{2\sigma_{\alpha,d}^2} \right) \tag{4.5}$$

$$k_2(\mathbf{f}_i, \mathbf{f}_j) = \exp\left(\sum_{d=(x,y,z)} \frac{|p_{i,d} - p_{j,d}|^2}{2\sigma_{\beta,d}^2} - \sum_{d=(x,y,z)} \frac{|I_{i,c} - I_{j,c}|^2}{2\sigma_{\gamma,c}^2} \right) \tag{4.6}$$

The final version of DeepMedic developed in [1] and used very successfully in ISLES 2015 is constructed as an 11 layer dual-pathway CNN. It uses 8 convolutional layers with 3^3 kernels, two fully connected hidden layers with 1^3 kernels, and a final classification layer. All layers use a stride of 1 since any greater stride would result in down sampling the image, which is not desirable for segmentation tasks. The two pathways in DeepMedic operate upon the full resolution original image, and a copy of the image down sampled by a factor of 3 respectively. Both pathways have a final receptive field of 17^3, so the full resolution pathway is expected to learn and detect features from only a local perspective, whereas the lower resolution pathway is able to learn and detect features from a much larger area, since it is able to sample the equivalent of a 51^3 volume from the original image.

4.2.2 IMPLEMENTATION

The DeepMedic code base for Linux systems has been made freely available online at [12], which also provides a set of default configuration files identical to those used for the segmentation of brain tumors for BRATS as described in [1]. Several DeepMedic models were trained using those configurations with only minor adjustments necessary for segmenting stroke images, as described in Glocker et al's ISLES report [13].

The CNNs trained for this experiment were all similar to the model used during [13]. They consisted of 8 convolutional layers, 2 fully connected layers, and 1 classification layer. The 8 convolutional layers used 30, 30, 40, 40, 40, 40, 50, and 50 feature maps respectively, all with 3^3 kernels. All models were trained for only 2 output classes since the ground truth labels provided with the ISLES 2015 data was binary, with 1 representing lesioned tissue, and 0 representing healthy tissue or image background. All layers used a Rectified Linear Units activation function. The low resolution pathway was identical to the full resolution pathway with the input down sampled by a factor of 3. To avoid overfitting to the training data, the final two layers had a dropout rate of 50%.

DeepMedic is a very resource intensive algorithm, especially during the training phase, as a result it was not practical to train these models using the CPU of a standard computer. The models were instead trained on an Ubuntu workstation provided by the LRDC which has an NVIDIA

GeForce GTX Titan X graphics card, that was used to run DeepMedic. Compiling a model on that card takes approximately 5 minutes, training that model then takes about 17 hours, then once trained it takes 2-3 minutes to segment a single patient's imaging. A total of 7 models were created and trained in order to compare the performance changes with various combinations of imaging modalities. One model was trained with each of the following combinations: T1/T2/FLAIR, T1/T2, T1/FLAIR, T2/FLAIR, T1, T2, and FLAIR. All model configurations were held constant between models except for the number of input channels, which must be changed according to how many modalities that model will be trained with.

DeepMedic models were trained using the training dataset from ISLES 2015. The ISLES training dataset contains 28 patients with T1, T2, DWI, and FLAIR images, 3 were selected at random to be withheld for validation while training the models. Once each of the 7 models was trained, they were used to segment all patients that each model could be applied to, shown in Table 2.

Table 2: Patients segmented using each DeepMedic model

Model	Censa Number of Patients Segmented
T1/T2/FLAIR	204, 214, 217, 221
T1/T2	202, 203, 204, 206, 208,214, 215, 217, 221
T1/FLAIR	201, 204, 205, 207, 210, 212, 214, 216, 217, 219, 220, 221
	222, 223, 224, 225, 227, 228, 229, 303, 304, 306, 307
T2/FLAIR	204, 214, 217, 221
T1	All
T2	202, 203, 204, 206, 208,214, 215, 217, 221
FLAIR	201, 204, 205, 207, 210, 212, 214, 216, 217, 219, 220, 221
	222, 223, 224, 225, 227, 228, 229, 303, 304, 306, 307

According to [1] there are four requirements that the input images must meet before they can be segmented successfully using DeepMedic.

1. Convert all DICOM series to Neuroimaging Informatics Technology Initiative (NIfTI) files
2. Remove all non-brain tissue from the images (Skull Stripping)
3. Resample to isotropic 1 mm^3 resolution
4. Normalize each image to zero mean and unit variance

Requirement 1 was achieved using the freely available MATLAB package found in [14]. The LRDC was able to manually skull strip each of the T1 images, and provide a brain mask which was used to strip the other modalities through a simple elementwise matrix multiplication in MATLAB to satisfy requirement 2. All of the input images supplied by the LRDC were already in 1 mm^3 resolution, so no further processing was necessary for requirement 3. Finally, requirement 4 was performed using a simple MATLAB script.

After segmenting the chronic stroke dataset from the LRDC, the same trained DeepMedic models were also used to segment the testing dataset from ISLES 2015. The testing dataset contains 36 acute stroke lesion images with all 3 modalities, meaning that all 7 models could be applied successfully to every patient.

4.3 GENERATIVE MODEL

In [2], Menze et al. develop a generative probabalistic model for lesion segmentation. This method was designed and tested using multimodal data containing T1, T1c, T2, and FLAIR images. Later, in [7], they develop discriminative extensions for that model that seek to improve accuracy by including regional pattern data that can't be accounted for in the basic generative model.

4.3.1 THEORY

Menze et al. first build a statistical model of the brain, which the generative model uses for automatic segmentation in [2]. The normal state of healthy tissues is modeled using healthy brain atlases for the three main tissue classes, Gray Matter, White Matter, and CSF. At each voxel i, the atlas is assumed to be a multinomial random variable for the tissue class k_i according to Equation 4.7, which is assumed to be true for all channels.

$$p(k_i = k) = \pi_{ki} \tag{4.7}$$

The lesion state is assumed to exist as a hidden atlas α, this atlas defines α_i, which is the probability that voxel i contains a lesion. They define a tumor state vector $\mathbf{t}_i = [t_i^1, ..., t_i^C]^T$, where each t_i^c is a Bernoulli random variable with parameter α_i that indicates whether a lesion is present in voxel i of channel c, and has the probability given by Equation 4.8

$$p(\mathbf{t}_i; \alpha_i) = \prod_c p(t_i^c; \alpha_i) = \prod_c \alpha_i^{t_i^c} \cdot (1 - \alpha_i)^{1 - t_i^c} \tag{4.8}$$

Next, image observations are generated by Gaussian distributions for each of the K classes and C channels, using mean μ_k^c and variance v_k^c. If a tumor is present in a given voxel, the Gaussian observation is replaced using another set of channel-specific Gaussian distributions using mean μ_{K+1}^c and variance v_{K+1}^c. If θ is defined as the set of all μ and v, and $\mathbf{y}_i = [y_i^1, ..., y_i^C]^T$ is a vector of observations at voxel i, the data likelihood is calculated using Equation 4.9.

$$p(\mathbf{y}_i | \mathbf{t}_i, k_i; \theta) = \prod_c p(y_i^c | t_i^c, k_i; \theta) = \prod_c [\mathcal{N}(y_i^c; \mu_{k_i}^c, v_{k_i}^c)^{1 - t_i^c} \cdot \mathcal{N}(y_i^c; \mu_{K+1}^c, v_{K+1}^c)^{t_i^c}] \tag{4.9}$$

The product of Equations 4.7, 4.8, and 4.9 is the joint probability of the tumor atlas and the observed variables as shown in 4.10.

$$p(\mathbf{y}_i, \mathbf{t}_i, k_i; \theta, \alpha_i) = p(\mathbf{y}_i | \mathbf{t}_i, k_i; \theta) \cdot p(\mathbf{t}_i; \alpha_i) \cdot p(k_i) \tag{4.10}$$

This model has two parameters, θ and α which need to be estimated before an image can be segmented. [2] proposes finding a maximum likelihood estimate of those parameters using Equation 4.11.

$$\langle \hat{\theta}, \hat{\alpha} \rangle = \arg\max_{\langle \hat{\theta}, \hat{\alpha} \rangle} p(\mathbf{y}_1, ..., \mathbf{y}_N; \theta, \alpha) = \arg\max_{\langle \hat{\theta}, \hat{\alpha} \rangle} \prod_{i=1}^{N} p(\mathbf{y}_i; \theta, \alpha) \tag{4.11}$$

Where N is the total number of voxels and $p(\mathbf{y}_i; \theta, \alpha) = \sum_{\mathbf{t}_i} \sum_{k_i} p(\mathbf{y}_i, \mathbf{t}_i, k_i; \theta, \alpha)$. The maximization is calculated using an iterative Expectation Maximization technique. Defining $\tilde{\theta}$ and $\tilde{\alpha}$ as the current estimates, the posterior probability of a lesion in voxel i is defined as Equation 4.12, and the posterior probability of finding class k at voxel i is defined as Equation 4.13.

$$q_i(\mathbf{t}_i) = p(\mathbf{t}_i | k_i, \mathbf{y}_i; \tilde{\theta}, \tilde{\alpha}) \propto \sum_{k_i} p(\mathbf{y}_i | \mathbf{t}_i, k_i; \tilde{\theta}) p(k_i) \tag{4.12}$$

$$w_{ik}(\mathbf{t}_i) = p(k_i | \mathbf{t}_i, \mathbf{y}_i; \tilde{\theta}, \tilde{\alpha}) \propto \pi_{ki} \prod_{c} \mathcal{N}(y_i^c; \tilde{\mu}_k^c, \tilde{v}_k^c)^{1-t_i^c} \tag{4.13}$$

Using the healthy and lesion posteriors the estimates of all parameters can be calculated for the next iteration using Equations 4.14 though 4.18. The posterior probabilities q_i and w_{ik} and the estimates $\tilde{\theta}$ and $\tilde{\alpha}$ are computed iteratively until they converge upon the final estimates of $\hat{\theta}$ and $\hat{\alpha}$, which are then used to segment the image.

$$\tilde{\alpha}_i \leftarrow \sum_{\mathbf{t}_i} q_i(\mathbf{t}_i) (\frac{1}{C} \sum_{c} t_i^c) \tag{4.14}$$

$$\tilde{\mu}_k^c \leftarrow \frac{\sum_i \sum_{\mathbf{t}_i} q_i(\mathbf{t}_i) w_{ik}(\mathbf{t}_i)(1 - t_i^c) y_i^c}{\sum_i \sum_{\mathbf{t}_i} q_i(\mathbf{t}_i) w_{ik}(\mathbf{t}_i)(1 - t_i^c)} \tag{4.15}$$

$$\tilde{v}_k^c \leftarrow \frac{\sum_i \sum_{\mathbf{t}_i} q_i(\mathbf{t}_i) w_{ik}(\mathbf{t}_i)(1 - t_i^c)(y_i^c - \tilde{\mu}_k^c)^2}{\sum_i \sum_{\mathbf{t}_i} q_i(\mathbf{t}_i) w_{ik}(\mathbf{t}_i(1 - t_i^c)} \tag{4.16}$$

$$\tilde{\mu}_{K+1}^c \leftarrow \frac{\sum_i \sum_{\mathbf{t}_i} q_i(\mathbf{t}_i) t_i^c y_i^c}{\sum_i \sum_{\mathbf{t}_i} q_i(\mathbf{t}_i) t_i^c} \tag{4.17}$$

$$\tilde{v}_{K+1}^c \leftarrow \frac{\sum_i \sum_{\mathbf{t}_i} q_i(\mathbf{t}_i) t_i^c (y_i^c - \tilde{\mu}_{K+1}^c)^2}{\sum_i \sum_{\mathbf{t}_i} q_i(\mathbf{t}_i) t_i^c} \tag{4.18}$$

Once an estimate of the model parameters is reached, the probability that channel c of voxel i is computed by summing over all $\mathbf{t}_i$ where $t_i^c = 1$, yielding Equation 4.19, and assigning it to tumor if $p(t_i C = 1|\mathbf{y}_i; \hat{\theta}\hat{\alpha}) > 0.5$, and to healthy tissue otherwise, resulting in a binary voxel segmentation.

$$p(t_i C = 1|\mathbf{y}_i; \hat{\theta}\hat{\alpha}) = \sum_{\mathbf{t}_i} t_i^C p(\mathbf{t}_i|\mathbf{y}_i; \hat{\theta}, \hat{\alpha}) = \sum_{\mathbf{t}_i} t_i^C q_i(\mathbf{t}_i) \tag{4.19}$$

At this point, the iterative process would become memory intensive if calculated for every possible vector $\mathbf{t}_i$. To avoid this, Menze et al. implement a set of 4 rules for tumor voxels based on biological observations, and remove any $\mathbf{t}_i$ that violate those rules during each iteration. First, any vectors that contain CSF in one channel, and lesion in another are removed. They also require that edema visible in the T2 channel should also be visible in the FLAIR channel as well, and lesions visible in T1 and T1c must be entirely contained within lesions visible in the T2 and FLAIR channels.

The final biological constraint compares modifies the probability of lesion in the i^{th} voxel of a given channel by comparing the current average intensity $\tilde{\mu}_k^c$ where k in this case is the class for white matter, with the current observation intensity for that voxel and channel y_i^c. In the T1 channel, lesions are expected to be hyper-intense, and the probabilities are modified according to Equation 4.20. For the T1c, T2, and FLAIR channels, lesions are expected to be hypo-intense, and the probabilities are modified according to Equation 4.21.

$$q_i(t_i^c) = \begin{cases} p(t_i^c|\mathbf{y}_i; \tilde{\theta}, \tilde{\alpha}), & \text{if } y_i^c > \tilde{\mu}_k^c \\ 0, & \text{otherwise} \end{cases} \tag{4.20}$$

$$q_i(t_i^c) = \begin{cases} p(t_i^c|\mathbf{y}_i; \tilde{\theta}, \tilde{\alpha}), & \text{if } y_i^c < \tilde{\mu}_k^c \\ 0, & \text{otherwise} \end{cases} \tag{4.21}$$

4.3.2 IMPLEMENTATION

An implementation of the generative model in Python has been made available by Menze et al. at [15]. The published code does not contain spatial regularization via Markov Random Fields or the discriminative extensions mentioned in [7], which both serve to reduce the false positive rate. This

algorithm was not extremely resource intensive and was run on a consumer grade Linux PC using a quad-core CPU and 12 GB of memory. Segmentation of a single patient with 3 available modalities only takes about 3 minutes and uses approximately 2 GB of memory. This algorithm does not use a learning technique, so there is no lengthy training phase before segmentation can begin. This method was designed to be applied to multimodal data containing T1, T1c, T2, and FLAIR images, however it can accept an input of any subset of those. Patient images were segmented using each of the 7 possible combination of input channels: T1/T2/FLAIR, T1/T2, T1/FLAIR, T2/FLAIR, T1, T2, and FLAIR according to Table 3.

Table 3: Input data combinations possible for the Generative-Discriminative model

Input Channel Combination	Censa Number of Patients Segmented
T1/T2/FLAIR	204, 214, 217, 221
T1/T2	202, 203, 204, 206, 208,214, 215, 217, 221
T1/FLAIR	201, 204, 205, 207, 210, 212, 214, 216, 217, 219, 220, 221
	222, 223, 224, 225, 227, 228, 229, 303, 304, 306, 307
T2/FLAIR	204, 214, 217, 221
T1	All
T2	202, 203, 204, 206, 208,214, 215, 217, 221
FLAIR	201, 204, 205, 207, 210, 212, 214, 216, 217, 219, 220, 221
	222, 223, 224, 225, 227, 228, 229, 303, 304, 306, 307

In addition to the data summarized in Table 3 the generative model was also applied to the acute stroke datasets from ISLES 2015. Since the ISLES patients contain all three modalities each combination of input data can be used with every patient.

According to [15] there are three requirements that the input images must meet before they can be segmented using this algorithm: all DICOM images must be converted to NIfTI, the images must be skull stripped, and all images, masks, and atlas maps must be in the same reference space.

As before, DICOM files can be converted to NIfTI easily using the MATLAB package in [14]. Skull stripping was performed partially by the LRDC, then completed using the masks they provided. All images, masks, and atlas maps were transformed to the same reference space by cross-registering every image for a particular patient using the free software package Elastix, available at [16], and described by [17, 18].

4.4 HISTOGRAM BASED GRAVITATIONAL OPTIMIZATION ALGORITHM

The HGOA is an interesting algorithm, but has not been developed as robustly as some of the others so far. It has never been tested against either of the major contest datasets, and has been designed specifically for Diffusion Weighted images.

This approach can be broken down into two components. The first attempts to segment the MRI input using a histogram-based brain segmentation algorithm, and the second is an n-dimensional gravitational optimization algorithm which minimizes the difference between the number of segments found in the histogram-based algorithm and the desired number of segments.

4.4.1 HISTOGRAM-BASED BRAIN SEGMENTATION

The first component of HGOA is the Histogram-Based Brain Segmentation algorithm, which is broken down into the 7 steps shown in Figure 1 from [19].

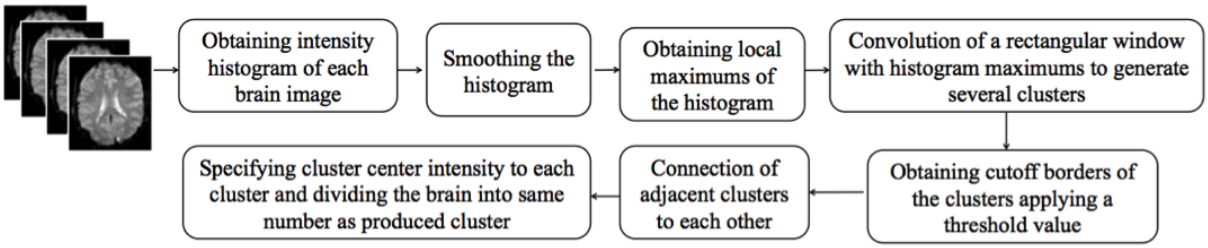


Figure 1: Flow chart of the Histogram-Based Segmentation Algorithm.

First the input image is processed with a low pass Gaussian filter. Then the image intensity histogram is calculated using Equation 4.22, where $H[n]$ denotes the normalized histogram of the image with a bin for each possible intensity, m_n is the number of voxels with intensity n, M is the total number of voxels in the image, and L is the number of possible intensity values.

$$H[n] = \frac{m_n}{M} \quad n = 0, 1..L-1 \tag{4.22}$$

Step 2 is to smooth $H[n]$ using a local weighted averaging technique, as described in Equation 4.23.

$$\bar{H}[n_i] = \sum_{i}^{i+G} w_i \cdot \frac{H[n_i]}{G} \tag{4.23}$$

$$w_i = \sum_{i}^{i+G} \|n_i - M_w\|^2 \tag{4.24}$$

Where $\bar{H}[n_i]$ is the local average value of the histogram, $H[n_i]$ is the histogram distribution value of the i^{th} bin, w_i is the weight of the i^{th} bin, and G is the length of the averaging window. The weights for each bin are calculated using Equation 4.24, where M_w is the average of the intensities in the window, and n_i is the voxel intensity of the i^{th} element. During this process, if G is increased the histogram will be smoother, and fewer local maxima will be found.

Step 3 is to locate the local maxima in the smoothed histogram using Equation 4.25.

$$H_{max}[n] = \bar{H}[n_i] | (\bar{H}[n_i] > \bar{H}[n_{i+1}]) \cap (\bar{H}[n_i] > \bar{H}[n_{i-1}]) \tag{4.25}$$

In step 4 the histogram local maxima from step 3, $H_{max}[n]$ is convolved with a rectangular window. This works upon the assumption that local intensity maxima correspond to unique brain structures that would appear as different intensities, such as gray matter, white matter, and CSF. Therefore, it is desired to automatically expand each local histogram maxima toward its neighbors using the convolution described in Equation 4.26.

$$Y[n] = Win[n] * H_{max}[n] = \sum_{j} Win[j] H_{max}[n-j] \tag{4.26}$$

Where W is the length of $Win[n]$ and M is the length of $H_{max}[n]$. This convolution will increase the width of each local maxima, and may result in maxima that are located close to each other by intensity may combine into a single maximum. Due to this, a wider $Win[n]$ will result in fewer segments than a very narrow window.

For step 5 the upper and lower cutoff thresholds are located. This process adds additional flexibility for the later optimization algorithm because increasing the threshold value T may decrease the number of detected segments by eliminating some lower maxima. Equations 4.27 and 4.28 define the lower and upper cutoff thresholds respectively:

$$X_{low}[n_i] = \{n | Y[n_{i+1}] > T \cap Y[n_{i-1}] < T\} \tag{4.27}$$

$$X_{high}[n_i] = \{n | Y[n_{i-1}] > T \cap Y[n_{i+1}] < T\} \tag{4.28}$$

Figure 2 demonstrates the effect that T has on the number of local maxima that are ultimately defined to be a unique segment. Using T_1 only a single segment is found, but using T_2 three segments are detected.

Step 6 defines the range of intensity values in the original image that are assigned to each detected segment. Every voxel in the image must be assigned to a segment, and once $Y[n]$ has been thresholded to determine the cutoff borders of each segment there may exist gaps between $X_{low}[s_i]$ and $X_{high}[s_{i+1}]$, where s_i denotes the i^{th} segment. These gaps are filled in by assigning those intensity values to either bordering segment proportionally according to Equation 4.29.

$$\begin{aligned} X_{high-new}[s_i] &= X_{high}[s_i] + (X_{low}[s_{i+1}] - X_{high}[s_i]) \times \frac{H_{max}[s_i]}{H_{max}[s_i] + H_{max}[s_{i+1}]} \\ X_{low-new}[s_{i+1}] &= X_{high-new}[s_i] \end{aligned} \tag{4.29}$$

Finally, step 7 defines a specific intensity value for each detected segment. To generate the final segmentation image, all voxels with intensity values between $X_{low-new}[s_i]$ and $X_{high-new}[s_{i+1}]$ in the original image will be converted to that respective segment's defined intensity, $X_{center}[s_i]$, calculated using Equation 4.30.

$$X_{center}[s_i] = \frac{X_{low-new}[s_i] + X_{high-new}[s_i]}{2} \tag{4.30}$$

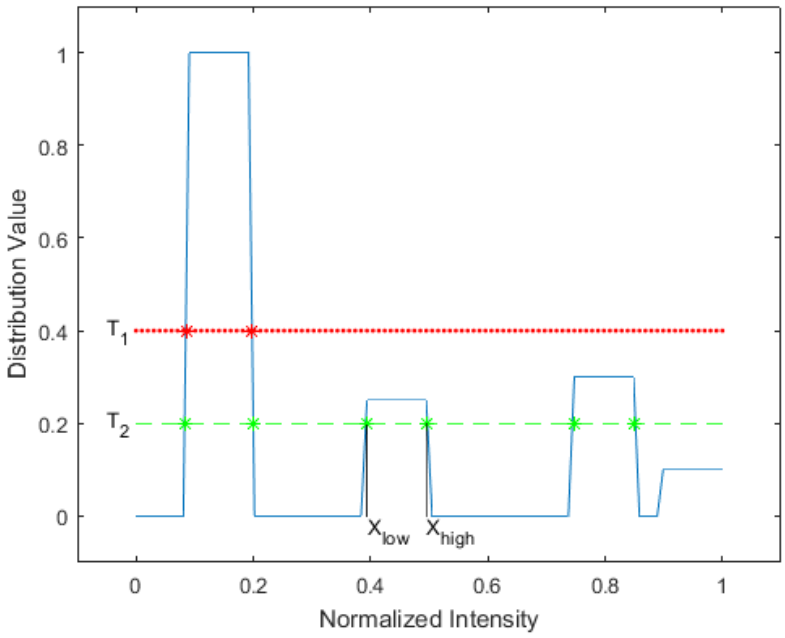


Figure 2: Example plot of $Y[n]$ with two threshold values

4.4.2 N-DIMENSIONAL GRAVITATIONAL OPTIMIZATION ALGORITHM

Each time the 7-step process described in section 4.4.1 is performed the resulting output will be different, and it is likely that a single iteration will not generate the desired number of segments, or even a usable segmentation. There are three adjustable parameters that will affect the number of segments generated in any single iteration. They are: the length of the averaging window G used in Equations 4.23 and 4.24, the length of the convolution window $Win[n]$ used in Equation 4.26, and the threshold value T used in Equations 4.27 and 4.28. Since the length of both windows can theoretically range from 1 to 256, and the threshold is defined on the interval $[0, 1]$ it is computationally impractical to calculate a significant subset of possible combinations, and some method to automate this process in an efficient manner is desirable.

The N-Dimensional Gravitational Optimization Algorithm (NGOA) is used to optimize an objective function defined as the squared difference between the desired and achieved number of segments. The theory behind NGOA is based on the principle of a gravitational field, functionally it is a simulation of K particles in space, and attempts to find the heaviest.

The particles are initialized by a random selection of K n-dimensional vectors, representing the position, velocity, and acceleration of those particles as described in Equations 4.31 through 4.33.

$$X_i = [[x_{i1}, x_{i2}, ..., x_{in}]^T \tag{4.31}$$

$$V_i = [v_{i1}, v_{i2}, ..., v_{in}]^T \tag{4.32}$$

$$a_i = [a_{i1}, a_{i2}, ..., a_{in}]^T \tag{4.33}$$

Given those parameters, the gravitational force acting upon the i^{th} particle is calculated using Equation 4.34.

$$F_i = \frac{\prod_{j \neq i} m_i \cdot m_j \cdot (K \times X_i(t) - \sum_{j \neq i} X_j(t))}{\sum_{j \neq i}((x_{i1} - x_{j1})^2 + ... + (x_{in} - x_{jn})^2) + I_\epsilon} \tag{4.34}$$

Where m_i is the inverse of the value of the objective function for the i^{th} particle since this is a minimization problem. Once the gravitational force has been calculated, the new velocity is calculated using Equation 4.35, and using the new velocity the new position is calculated using Equation 4.36.

$$V(t+1)_i = \frac{ga_i}{min(a_j|_{j=1:K})} + V(t) \tag{4.35}$$

$$X(t+1)_i = V(t+1)_i + X(t)_i \tag{4.36}$$

In this case, $N = 3$, corresponding to the three parameters of the histogram based segmentation algorithm. HGOA calculates NGOA iteratively, each time using the result to perform a histogram based segmentation before the next iteration. This process continues until the objective function is satisfied, the maximum number of iterations is reached, or the result of Equation 4.35 drops below a set threshold.

Once the NGOA has converged, a simple thresholding operation defined by Equation 4.37 is performed to identify the segments containing stroke lesion.

$$Lesion = [X_{low}[s_i] < q_1 \cap X_{high}[s_i] < q_2] \tag{4.37}$$

According to [19] q_1 and q_2 are defined as 2.25 and 4.85 respectively for tumor detection, and the lesion would always be located in s_2. It does not go into detail about whether these same values were used for segmenting stroke-containing MRI. For this experiment multiple values of q_1 and q_2 were tested, and the restriction to segment 2 was relaxed.

4.4.3 IMPLEMENTATION

The author of HGOA has not made their source code publicly available. Since there was no source code, a custom implementation was written using MATLAB according to the mathematical theory published in [19], and summarized in Sections 4.4.1 and 4.4.2. HGOA does not allow multiple input channels, so segmentations were only performed on each individual modality for every patient. Running the program on a single patient takes roughly 30 seconds, however the algorithm does not always converge in a single attempt. Typically it takes 1-2 minutes to generate a segmentation on a typical consumer-grade CPU.

Since a custom written implementation of HGOA was used, there were no required manual pre-processing steps. According to the figures in [3] it does not appear that Nooshin et al. were using skull-stripped data, so the original images including the skulls were input to this algorithm. The only remaining pre-processing performed was converting to NifTI, as they were simpler to work with, and that conversion was necessary for the other algorithms resulting in no additional processing time. The low pass Gaussian filter as described in [19] was applied to the input image as the first step of the program.

4.5 LINDA

Pustina et al. describe the entire process to design, train, and use LINDA in [4]. In that experiment, a set of 60 chronic stroke patients was used to train and test the algorithm, to a high degree of success. Later it was successfully applied to a secondary dataset of 45 cases from a different lab, made with a different scanner.

In [4] Pustina et al. propose a set of 12 features that can be calculated from each of the T1 images when developing LINDA. In the interest of computational efficiency, only the top 6 performing features were included in the final LINDA algorithm. To determine the highest perfoming features Pustina et al. use the following iterative process.

The input dataset of 60 patients was split into a training group and a test group, consisting of 48 and 12 patients respectively. Using the training group, a model was trained for each of the 12 proposed features, then that model was used to segment the 12 test patients. The Dice score was calculated for each segmentation, and the feature with the highest aggregate score was selected. In the next iteration only 11 models were trained, this time using the top performer from the first iteration paired with each of the remaining 11. Those models are again tested, and the top pair is selected, then the process repeats until no further improvement was observed. The final selected features are as follows:

1. **Subject Specific Asymmetry:** Computed by reflecting the image about the Y axis, then subtracting that from the original image.
2. **K-Means Segmentation:** Divides a collection into K groups by calculating the mean of each group, calculating the distance between each point and mean, then re-grouping the points based on which mean they are nearest to. This process is repeated until the sum squared error within each group stabilizes.
3. **Gradient Magnitude:** Calculated as $\|(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial z})\|$ where I is the volume image. Denotes the amount of intensity change in an image at each voxel.
4. **Deviance from Control:** The remaining three features are all calculated by subtracting the last three features, from the average of that feature calculated using a group of 80 healthy control images.

4.5.1 TRAINING

Pustina et al. use a multi-resolution voxel-neighborhood random forest algorithm (MRVNRF). To build the trained model, a series of random forest models are trained at progressively increasing resolutions. At each resolution step the random forest model is trained on a matrix containing some feature data from all subjects. Each row pertains to a single voxel from a particular patient, with information from each of the features arranged along the columns of that row. Random Forest training does not require information from every single voxel, so this matrix is constructed using a randomly selected subset of 200 voxels from each class. The ground truth for each training step is simply each voxel's binary lesion status, either healthy or lesion.

When training at a given resolution step is complete, the newly generated model is applied to the training subjects to generate two new features, posterior probability maps of lesion and healthy tissue. The next training step is the same as the first, at a higher resolution, with the posterior probability maps from the first step as additional features. This process of training, generating posterior probability maps, and then increasing resolution is repeated until it has been performed for the desired number of resolution levels.

Once training is complete, the trained model can be applied to segment new subjects using a very similar process to training. When segmenting, the posterior probabilities calculated at the end of each resolution step are calculated by applying the previously trained model. At the end of the highest resolution step the posterior probabilities are converted into a binary segmentation map, assigning all voxels to a class based on which has the highest posterior probability.

4.5.2 IMPLEMENTATION

Pustina et al. have made the source code for segmentation using LINDA available to the public online at [20]. This distribution comes with a model trained from a set of 60 chronic, left-hemispheric, stroke patients from the University of Pennsylvania with T1 images. As of time of writing source code for training new models using other, or multiple modalities had not been published online.

LINDA can be run on a typical consumer-grade CPU, however it does require a significant amount of RAM, and is only supported on Linux. Due to these necessities all segmentations using LINDA were performed using the high performance computing infrastructure available at

the University of Pittsburgh Center for Simulation and Modeling (SAM). Each segmentation was run on a SAM computation node with 4 processor cores and 8 GB of RAM. Processing time exhibited a high level of variance, for some patients no more than 1.5-2 hours was necessary, for others processing time increased to roughly 8 hours. At this time it is not clear if LINDA is able to take advantage of higher parallelization on more than 4 CPU cores for improved performance.

Since LINDA is only capable of taking a single input channel, a segmentation was attempted with individual modalities, but was only successful with T1 images. LINDA performs most typical pre-processing techniques such as skull-stripping and bias field correction automatically, however according to [4] there are two pre-requisites that must be satisfied prior to using LINDA for segmentation:

1. Input images must be converted to the NifTI format
2. Input images must either have a left hemispheric stroke, or if right-hemispheric be left-right flipped. Bilateral lesions can be expected to have poor results

Requirement 1 was satisfied by converting all DICOM images to NifTI using the MATLAB package available in [14]. All of the patients in this study had left hemispheric strokes, so no further processing was required to fulfill requirement 2.

4.6 ANALYSIS

The output segmentation from each algorithm is a volumetric image with binary intensities. In computational terms this is simply a 3-D matrix where every element (voxel) is either 1 or 0, with 1 denoting the lesion class, and 0 denoting the background class which includes all healthy tissue as well. Similarly, the manual segmentations provided by the LRDC are also binary-valued matrices that were segmented using the T1 images, resulting in a size of $192 \times 256 \times 256$.

Each algorithm works in a different reference space, DeepMedic and HGOA are not dependent on any particular reference space, therefore they can operate on the lesion containing images directly, once each modality is cross registered on a per-patient basis. Prior to processing each patient, the different modalities were registered to the T1 image using Elastix, resulting in an

output size of $192 \times 256 \times 256$. LINDA automatically transforms input images to a space of $181 \times 217 \times 181$, and provides its output in the same space. The generative model described in Section 4.3 requires a healthy atlas for white matter, gray matter, and CSF. The atlas provided has dimensions $160 \times 216 \times 176$, requiring that all patients are transformed down to that size prior to use, then provides an output of the same size. Prior to calculating the following statistics, copies of the manual traces were transformed to the same size as the output from both LINDA and the Generative model using the Transformix package in Elastix.

Most current literature, including proceedings from BRATS and ISLES, use the Sørensen-Dice Coefficient, or Dice Score (DSC) to gauge the performance of their algorithms. The DSC compares the similarity of two samples and will range from 0 to 1, with 1 being a perfect match. It is frequently used in any image segmentation problem since it gives a convenient measure of the overlap between two segmentations. DSC is calculated according to Equation 3.1. Calculating the DSC here allows direct comparison to the performance described in other papers, and serves as an intuitive measure of how successful each algorithm is.

Since segmentation of stroke lesions is a binary detection problem, each algorithm is essentially a receiver, and there are four other statistics that are useful for characterizing a binary receiver. Sensitivity (SEN) measures the proportion of positives that are correctly identified, Specificity measures the proportion of negatives that are correctly identified, Accuracy measures the total proportion of positive and negative results that are correct, and Precision (PRC) which measures the proportion of true positives to the total number of positive results.

A single MRI image used here contains over 12 million total voxels. The typical large stroke lesion will contain less than 500 thousand voxels, representing only 4% of the total image volume. Due to this proportion, the metrics specificity and Accuracy will both represent the performance of the algorithms very poorly since they will be greatly inflated by the large number of true negative results, so they will not be used.

Formulas for Sensitivity and Precision are given as Equations 3.2 and 3.4 respectively, where P_f is the number of false positive predictions, N_t is the number of true negative predictions, and N_f is the number of false negative predictions. For each patient, the ground truth segmentation and the output segmentation were read into MATLAB as 3-D matrices. Those matrices were first processed with the round function, to guarantee that all voxels were either a 1 or 0, then P_t, P_f, N_t, and N_f are calculated by counting the nonzero elements of the output matrices from the logical operations shown as Equations 4.38 through 4.41 respectively.

$$TP = S \wedge G \tag{4.38}$$

$$TN = S \uparrow G \tag{4.39}$$

$$FP = S \wedge \neg G \tag{4.40}$$

$$FN = \neg S \wedge G \tag{4.41}$$

Once the four basic cardinalities are computed, it is straightforward to calculate the Dice score, Sensitvity, and Precision using Equations 3.1, 3.2, and 3.4 respectively.

The ISLES 2015 competetion calculated all of the previous metrics, along with two additional ones which will be used here so that results can be compared directly, the Hausdorff Distance (HD) and Average Symmetric Surface Distance (ASSD). The Hausdorff distance between two finite sets A and B is defined by Equation 3.14 and the ASSD between an automatic segmentation S_A and manual segmentation S_M is calculated with Equation 3.19, both the Hausdorff Distance and the ASSD are more difficult to compute than the overlap based metrics, however since the manual lesion traces are not available for the ISLES 2015 acute stroke data those metrics cannot be calculated directly. Instead, the automatic segmentations were uploaded to the ISLES 2015 website, which calculates them automatically once the segmentations have been submitted.

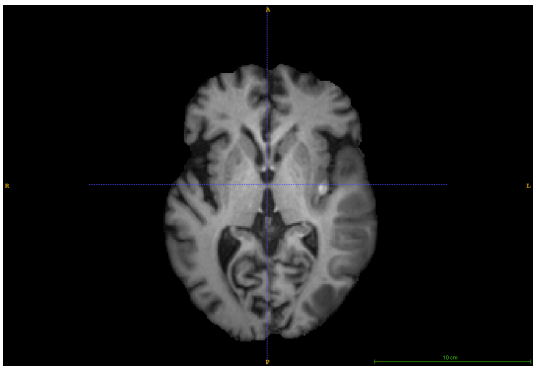
5.0 RESULTS

This chapter Presents numerical results of segmentations with each algorithim on each dataset, with some images of representative segmentations. Each of the performance metrics discussed in Section 4.6 have been calculated for cross algorithm comparisons. Deeper discussion of the results is found in Chapter 6.

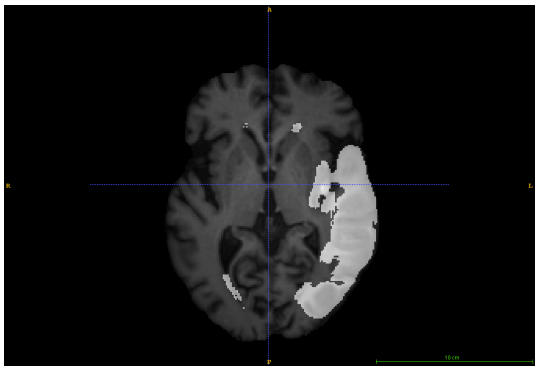
5.1 DEEPMEDIC

When a DeepMedic model is trained using the ISLES 2015 acute stroke training data, and used to segment the ISLES 2015 it generally performs well. Figure 3 provides a representative example of a very successful segmentation. Image (a) of Figure 3 shows the base image before segmentation, the lesion is most noticable along the edge of the lower right portion of the image, presenting as mostly gray intensities, most noticable due to the asymmetry with the left side of the image. Image (b) is the same slice of that patient's scan with an overlay of the DeepMedic generated segmentation. In the overlay, all of the highlighted voxels were classified as lesion by DeepMedic, and everything else was classified as background. This image shows that DeepMedic detected the majority of the lesion, with only a few small regions with what are likely to be false positives. The ISLES organizers will not release the individual manual segmentations so it is difficult to be certain if they are small lesions or false positives, however since this segmentation achieved a Dice score of 0.9 we know it was very accurate.

A representative example of a very poor segmentation for all combinations is also proviced in Figure 4, all trained models acheived a Dice score of 0 for this patient. In this case, the stroke lesion appears to be located in the lower left hemisphere of the image, but presents with mostly darker



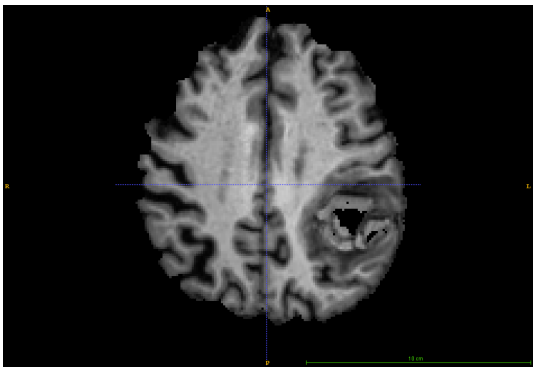
(a) Base Image



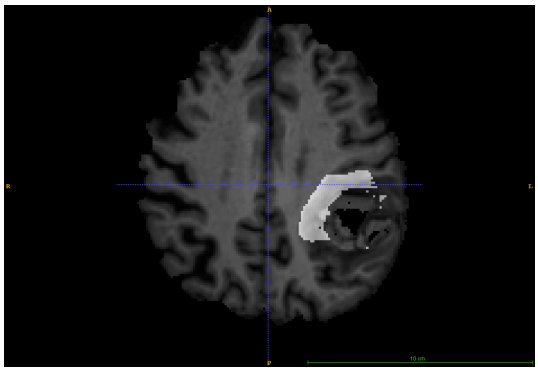
(b) With Lesion Overlay

Figure 3: Example of DeepMedic generated segmentation on SISS patient 2, DSC = 0.9.

voxel intensities than the lesion in Figure 3. Since the manual segmentations from ISLES 2015 are not publicly available it is impossible to be certain what voxels are considered lesion in the online evaluation tool. This patient's images may even be partially corrupted, as there is a small section in the middle of what looks like lesion that has voxel intensities of exactly zero, which would be very unusual for a medical scan.



(a) Base Image



(b) With Lesion Overlay

Figure 4: Example of DeepMedic generated segmentation on SISS patient 36, DSC = 0.

Figure 5 shows the average Dice scores acheived by applying DeepMedic to the SISS data. Testing with the acute SISS data showed that there was a decline in performance as input channels were removed. The far left bar in Figure 5 labeled "ISLES" is the result that Glocker et al entered into ISLES 2015 and reported in [13] using 4 input modalities, all of the 3, 2, and 1 channel models tested here performed worse than the 4 channel model, however there also appears to be some dependance on the FLAIR channel. In these tests, the average Dice score of 0.37 for the three channel model that used T1, T2, and FLAIR images was equivalent to the other three models that included the FLAIR modality. Models that did not include the FLAIR modality performed strictly worse, with an average Dice score of 0.27 for the two channel model using T1 and T2 images, then 0.22 and 0.16 for the T2 and T1 only channels respectively.

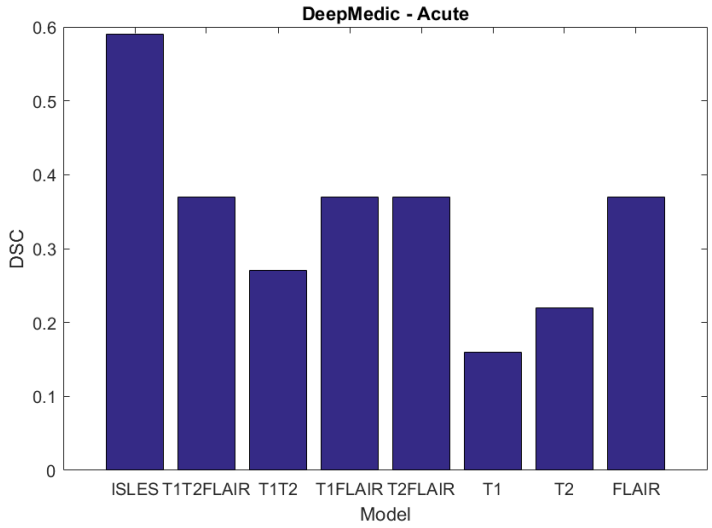
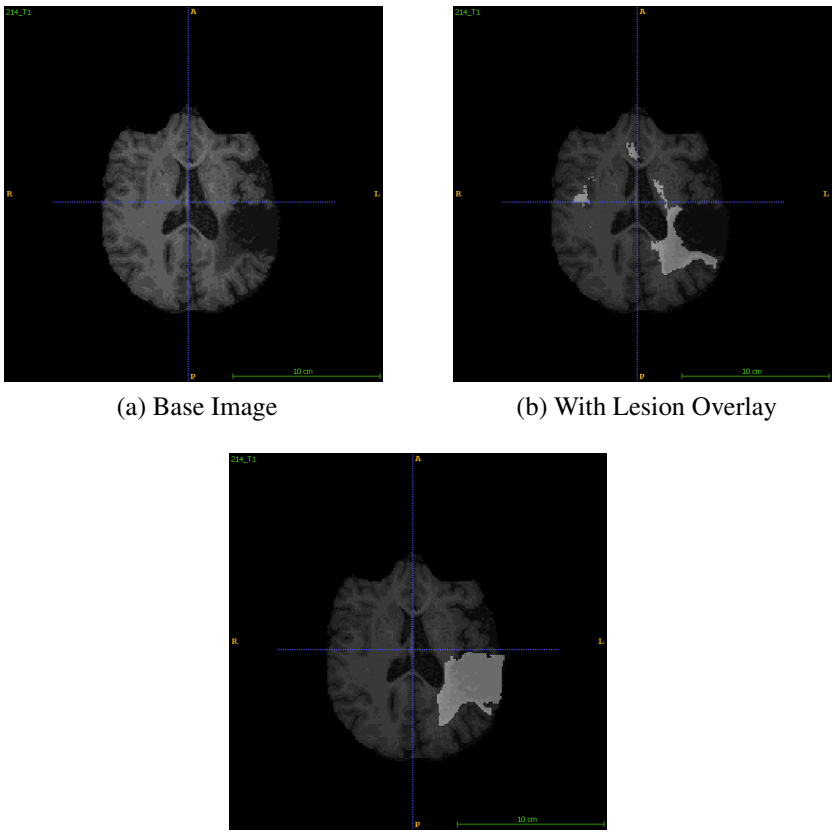


Figure 5: Average Dice Scores for DeepMedic using each combination of acute input data.

When DeepMedic was trained using the ISLES 2015 acute stroke training data, then used to segment the chronic stroke data supplied by the LRDC average Dice scores are significantly lower then when the training and segmentation data sets both contained acute stroke images. Figure 6 shows a typical segmentation of a chronic stroke by the three channel DeepMedic model. Image (a) is the base image of that patient's scan, image (b) is the same slice with overlay of the segmentation generated by the three modality DeepMedic model. As before, the highlighted voxels in (b) were classified by DeepMedic as lesion, and the rest were classified as background. In this case we have

access to the expert manual segmentation, presented as the overlay in image (c), and we can see that even when trained with the mismatched acute data, DeepMedic is able to recognize some of the stroke damage that appears as lighter voxel intensities in the gray range, but completely misses the large black area that is obvious to a human observer.



(a) Base Image

(b) With Lesion Overlay

(c) True Lesion

Figure 6: Representative segmentation of a chronic stroke by the 3 channel DeepMedic model trained with acute data

Individual Dice scores for each acute image using all models are shown in Figures 7 through 9. With each model, DeepMedic segmentations displayed a high degree of variance. In all cases, there were some patients that completely failed to segment, and some for which Dice scores of 0.8 or greater were acheived. Generally, between models the patients that DeepMedic completely failed to segment remained the same, indicating that some stroke lesions may not present features that DeepMedic is able to learn from the training dataset.

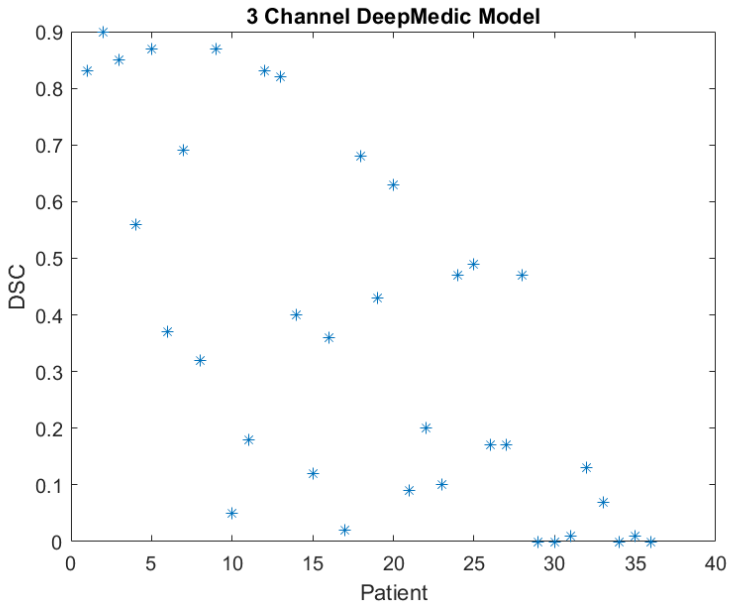


Figure 7: Individual Dice scores for all DeepMedic segmentations using the three modality model.

Similar trends were present in all metrics that were investigated. Figure 10 shows the average Sensitivity that DeepMedic acheived when applied to the matched acute/acute training and segmentation sets. In Figure 10 the far left bar is again the result published by Glocker et al in [13] when using their four modality model on the acute ISLES 2015 data. There is generally a decline in performance as input modalities are removed, and models containing the FLAIR images perform strictly better than models without them, in this case the two modality T1/FLAIR and T2/FLAIR models even have a slightly higher sensitivity than the three modality model by a small margin of 0.03 and 0.04 respectively. Since the Dice score is conveinient and easy to understand it will be used througout the remainder of this chapter.

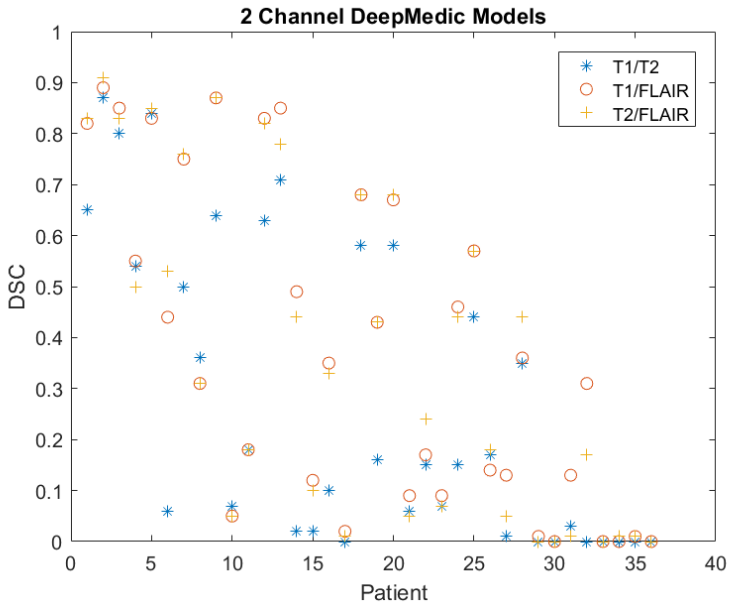


Figure 8: Individual Dice scores for all DeepMedic segmentations using the two modality models.

Figure 11 shows the average Dice score acheived by DeepMedic when segmenting the chronic stroke images. The same general trend was present that as input image modalities were removed, performance suffered with the exception of the T1 only case. In this experiment, there no longer appeared to be a strong dependance on the FLAIR modality as there was in the acute case. Now the T1 images appear to be most significant, with both T1-containing two modality models outperforming tthe T2/FLAIR model and the single modality T2 or FLAIR models. The T1-only model performed suprisingly well, acheiving an average Dice score of 0.32, higher than any of the other models when segmenting chronic data.

Figure 12 shows a segmentation of the same patient as in Figure 6, but using T1-only model, which was the top performing model when the training and test data were mismatched. We can see here that this model still detects most of the lesion area that the three modality model found, but was also able to pick up some of the black region that the three modalitiy model missed completely.

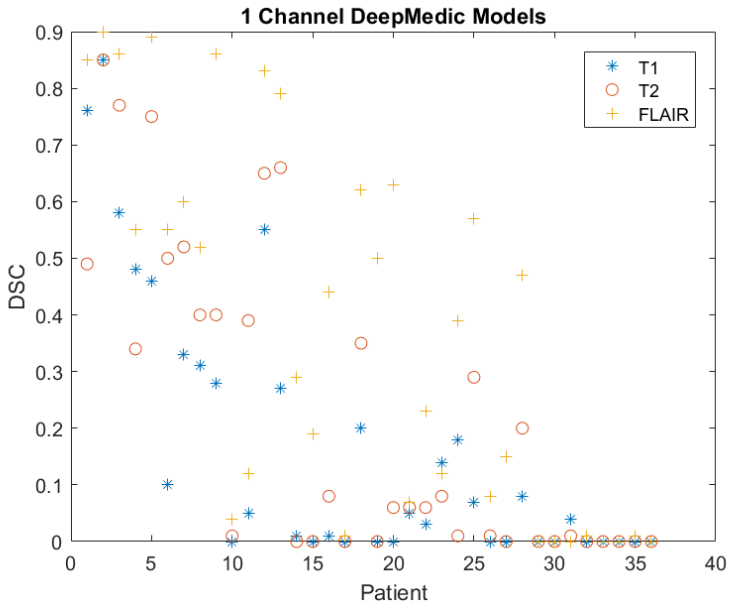


Figure 9: Individual Dice scores for all DeepMedic segmentations using the single modality models.

Figure 13 shows the individual results for each chronic stroke patient using the top performing T1 only DeepMedic model. Similarly to the results with the acute stroke dataset, the individual results show a high rate of variance. Several segmentations are very successful, with Dice scores in the 0.6-0.8 range, but many completely failed to segment with Dice scores very close to 0.

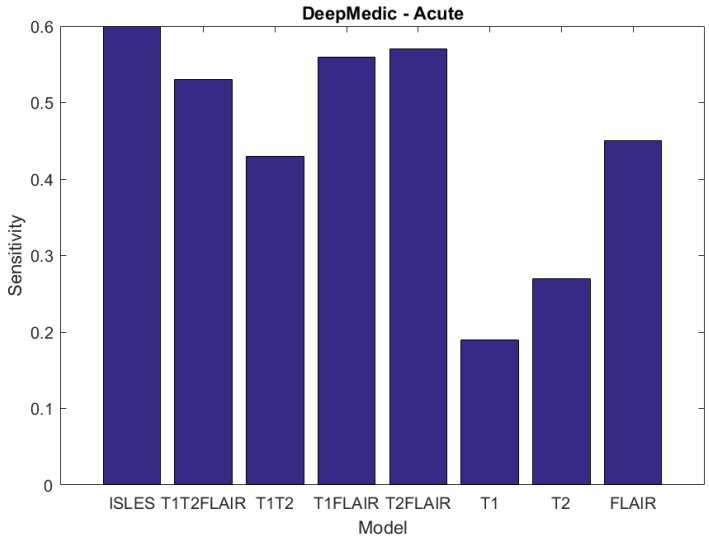


Figure 10: Average Sensitivity for DeepMedic acute segmentations.

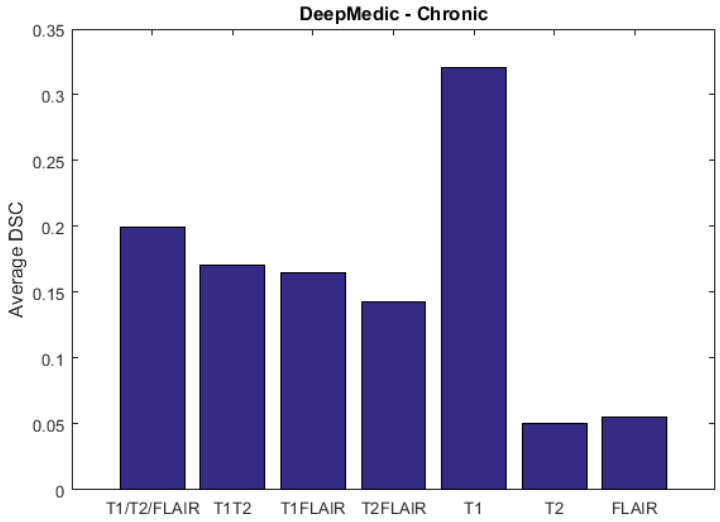
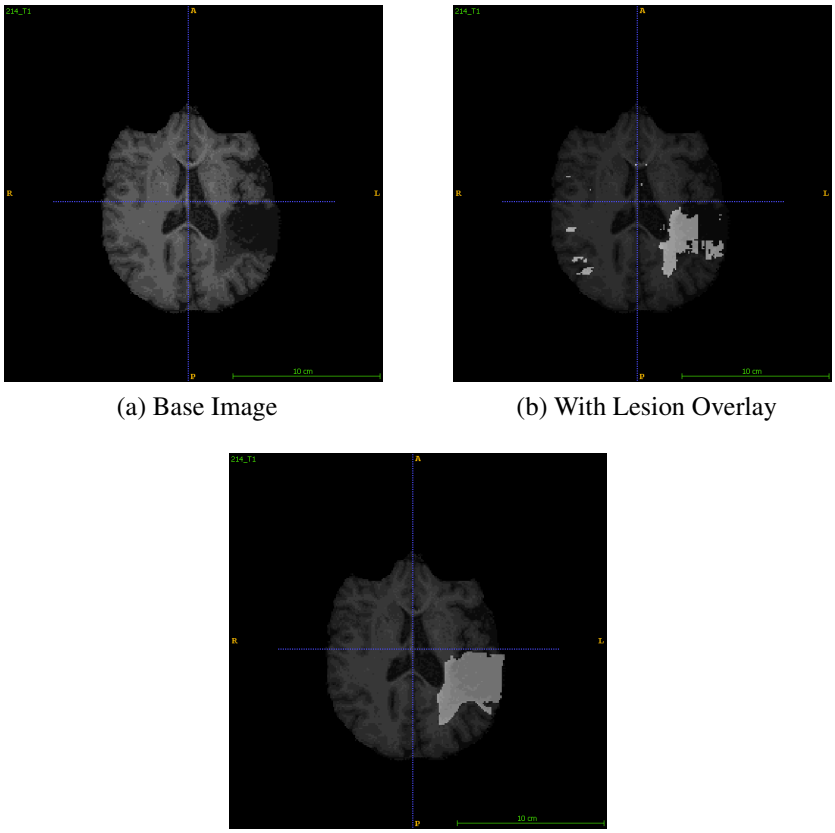


Figure 11: Average Dice Scores for DeepMedic using each combination of chronic input data.



(a) Base Image

(b) With Lesion Overlay

(c) True Lesion

Figure 12: Representative segmentation of a chronic stroke by the T1-only DeepMedic model trained with acute data

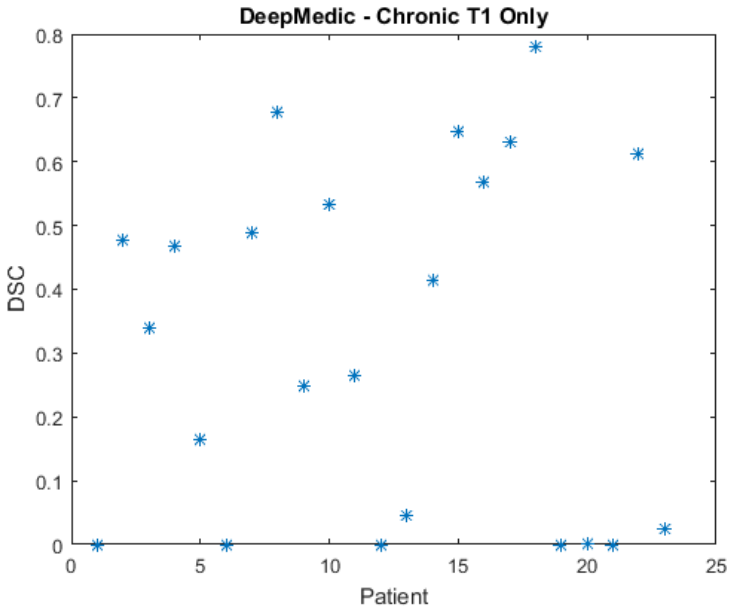
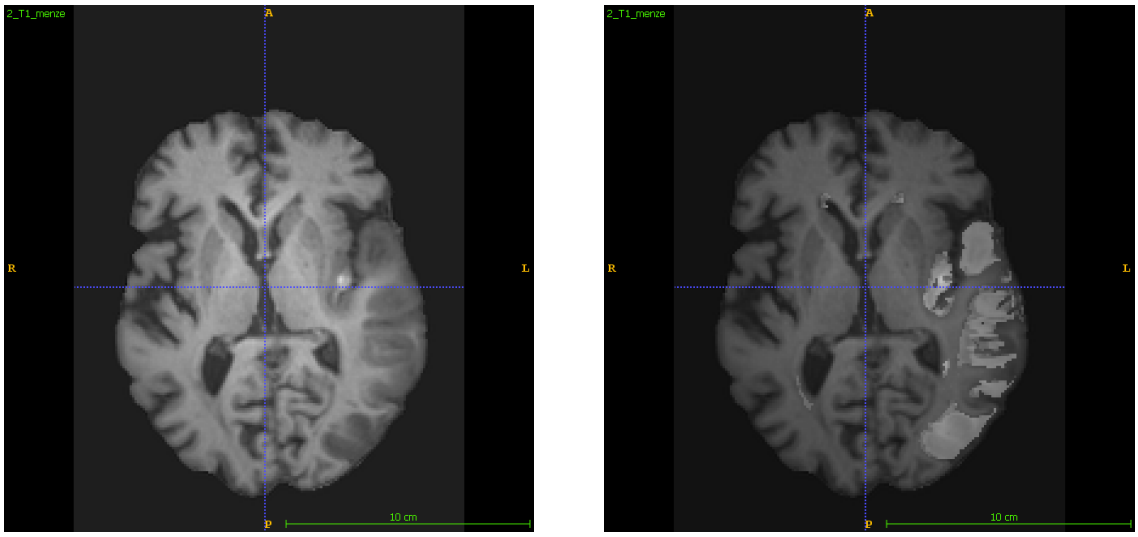


Figure 13: Individual chronic segmentation results using the T1 only DeepMedic model.

5.2 GENERATIVE MODEL

The generative model was originally designed for acute strokes, and tested against an acute stroke dataset in [7]. Here, it was first tested using the same acute stroke dataset that was used for DeepMedic in section 5.1. An example segmentation by the generative model is shown in Figure 14, using the same patient as Figure 3 so that the two can be compared directly. Compared to Figure 3 we can see that most of the lesion is still captured, however the generative model misses some parts of it that present with lighter voxel intensities closer to white. Similarly to DeepMedic, there were several patients that were unable to be segmented by the generative model at all. In some cases the algorithm actually diverges and crashes, and in other cases it fails to generate any meaningful segmentation. When applied to SISS patient number 36, the same one pictured in Figure 4 the generative model runs successfully, but the output segmentation has no voxels classified as lesion, resulting in a Dice score of 0.



(a) Base Image (b) With Lesion Overlay

Figure 14: Representative segmentation of an acute stroke with the generative mode, DSC = 0.61.

When applied to this acute data, the generative model underperformed the results reported in [7], the average results are shown in Figure 15. However, this experiment was performed with a different dataset entirely and used only up to 3 input modalities: T1, T2, and FLAIR as no T1c

images were available. The same general trends that were present in Section 5.1 are again present with the generative model. As input modalities are removed, the performance of the generative model suffers and the FLAIR images appear to be the most significant individually. Both of the FLAIR-containing two modality tests outperform the T1/T2 set, and the single modal test with just FLAIR performs nearly as well as the T2/FLAIR and three modality tests, outperforming both the T1/FLAIR and T1/T2 tests.

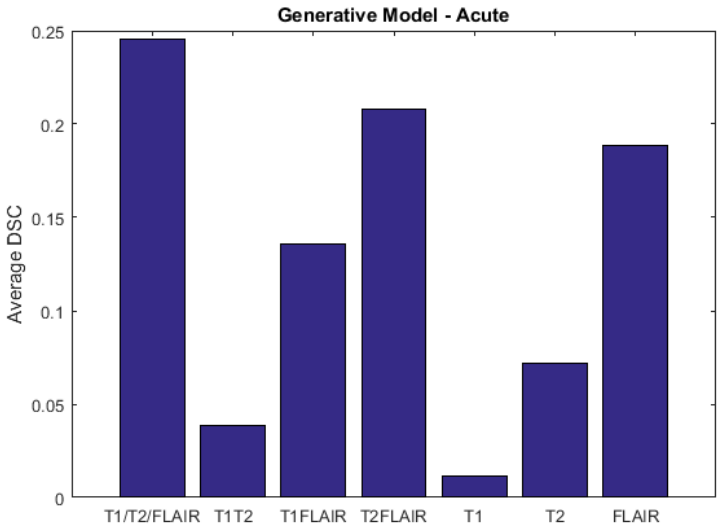
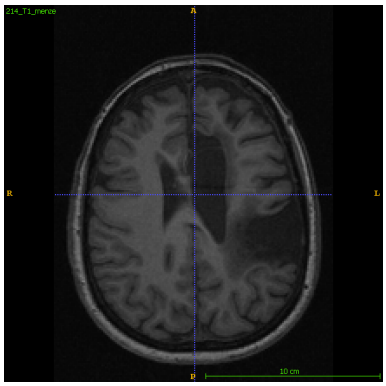


Figure 15: Average Dice Scores for the generative model applied to acute data.

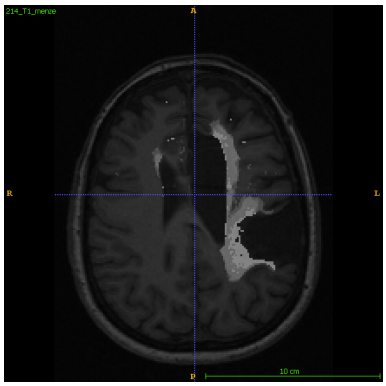
When applied to the three channel chronic stroke data, the generative model's performance generally declined. Figure 16 contains a segmentation created using the generative model of the same patient used in Figure 6. Similarly to the segmentation of this patient using DeepMedic, the generative model is able to pick up a portion of the edge of the lesion where the voxels have higher intensities in the gray range, but misses the large black region. Figure 16 was segmented using the generative model given three modalities: T1, T2, and FLAIR, but is similar to the segmentations with each other subset of two or one input modality.

The average Dice scores for segmentations with the generative model using each subset of chronic input modalities are shown in Figure 17. In this experiment there was no clear trend of declining performance as input modalities were removed as there was with the acute stroke data. However, the performance was generally much worse in this case since this algorithm was designed

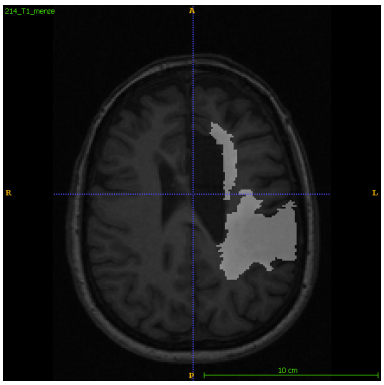
for acute stroke images, and we do not have any T1c data. Between the different combinations of input data there was a fairly small variation in performance, the top performing T2 model achieved an average Dice score of 0.19, and the worst performing T2/FLAIR model an average of 0.09.



(a) Base Image



(b) With Lesion Overlay



(c) True Lesion

Figure 16: Segmentation of Censa 214 with the 3 channel generative model.

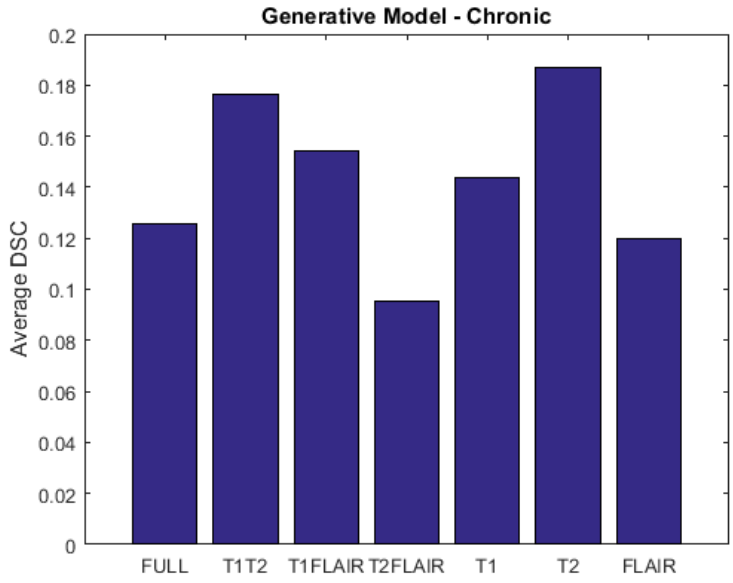


Figure 17: Average Dice Scores for the generative model applied to chronic data.

5.3 LINDA

LINDA was able to segment most of the chronic stroke data successfully, however was only capable of accepting T1 images as an input. The average dice score of the LINDA segmentations was 0.44, lower than the 0.67 acheived in [4] when LINDA was applied to a different lab's images. Figure 18 shows the Dice score for each individual segmentation, demonstrating that as with the previous algorithms, many of the individual segmentations had much higher Dice scores, but there were several individual patients that it completely failed to segment, decreasing the average performance.

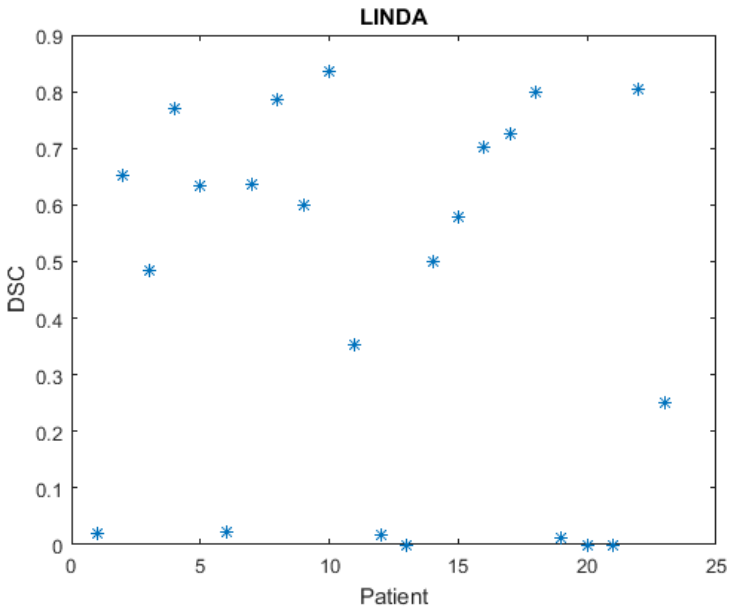
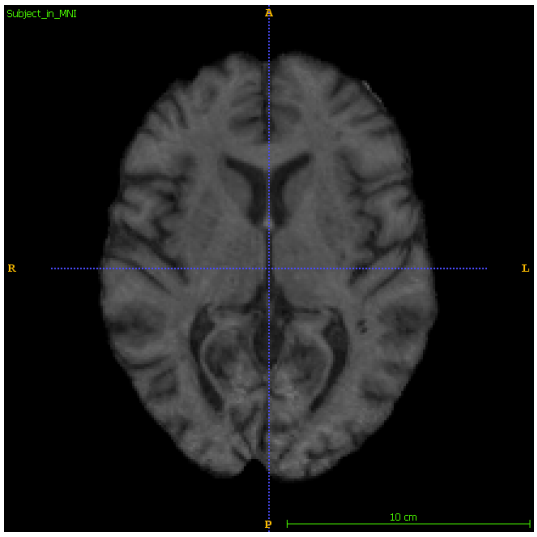
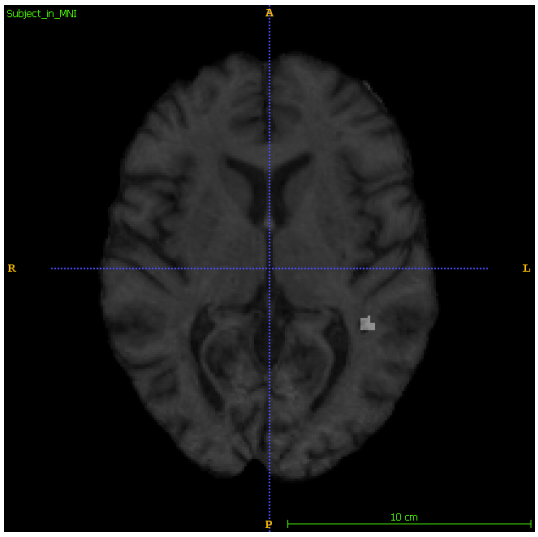


Figure 18: Dice scores for each individual patient's segmentations with LINDA

The images that LINDA failed to segment contained only very small lesions, a representative example of one such patient overlayed with the manual tracing is shown in Figure 19. In this particular case, the output segmentation that LINDA generated did not contain any voxels classified as lesion. In image (b) of Figure 19 only a small area is classified as lesion, the cluster of highlighted voxels in the lower left hemisphere of the brain.



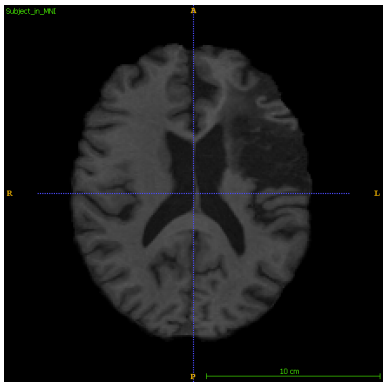
(a) Base Image



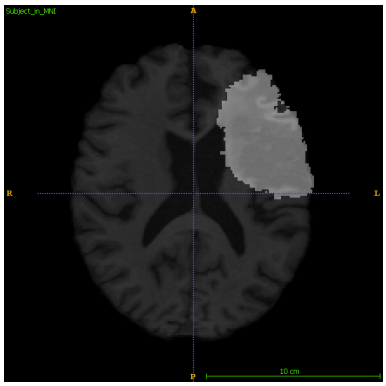
(b) True Lesion

Figure 19: Censa 304 image overlayed with manual segmentation, LINDA did not detect any lesion voxels.

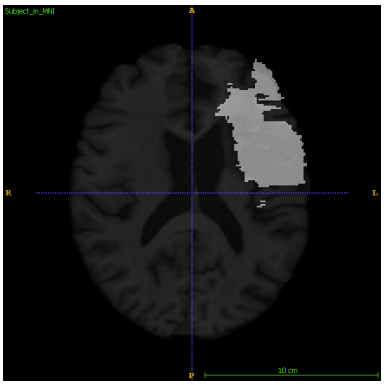
Figure 20 is representative of larger lesions that were segmented correctly. The publicly released model available at [20] was trained entirely with T1 images of chronic stroke patients, and as a result LINDA does not suffer from the same issue that DeepMedic and the generative model did where it misses legion voxels that appear as black intensities. In image (b) of Figure 20 we can see that LINDA identified nearly the entire lesion as shown in image (c). There is a small lesion area in the lower left hemisphere separated from the large one that LINDA misses entirely, and there are some false positive regions in the large continuous lesion in the upper left hemisphere. In this example LINDA achieved a Dice score of 0.8, reduced mostly by the false positive regions.



(a) Base Image



(b) With Lesion Overlay



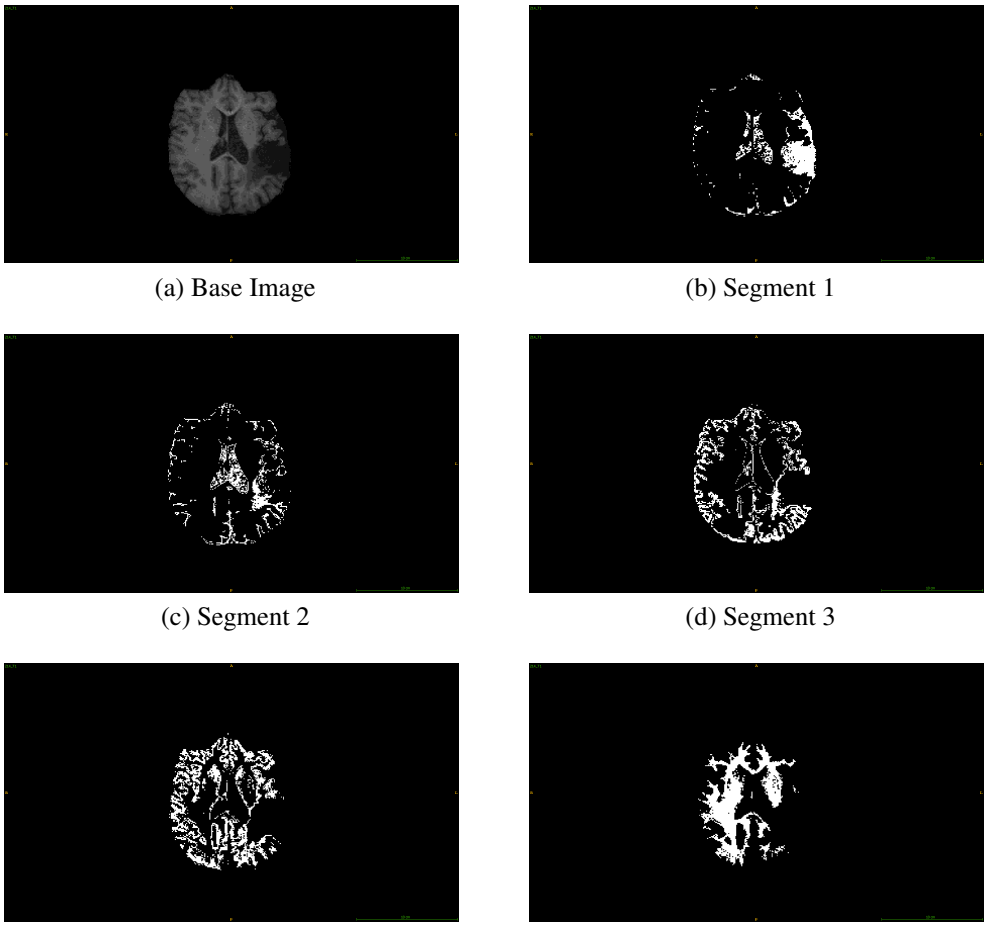
(c) True Lesion

Figure 20: Segmentation of Censa 306 with LINDA.

5.4 HGOA

The HGOA algorithm was not able to produce any useful segmentations using either the chronic or acute datasets. In [19] values of 2.25 and 4.85 for q_1 and q_2 in Equation 4.37 were used to detect a lesion segment. In testing the algorithm, generated segments were never able to satisfy that requirement, and no other values for q_1 and q_2 that worked in general could be identified. Looking at many of the segmentations it was apparent that it would be difficult for any available patients to meet these criteria since the lesions typically spanned a larger range of intensity values.

An example segmentation of a chronic stroke patient using HGOA is provided in Figure 21. In this example, HGOA was run with a target of 5 segments, and none of the resultant segments satisfied Equation 4.37 using the values given in [19]. Image (a) in Figure 21 shows the original T1 scan, and images (b) through (f) show the five unique segments that the HGOA algorithm identified. In this example, some lesion voxels are distributed between all 5 segments; most of the lesion is contained in segments 1, 2, and 3, with only small portions being identified in segments 4 and 5. While segment 1 contained the single largest concentration of lesion voxels, without finding some general criteria that satisfy Equation 4.37 there is no objective way to automatically classify any segment as lesion. Selecting segment 1 as lesion in this case would in fact be a violation of another restriction detailed in [19] that requires the lesion to always be located in segment 2.



(a) Base Image

(b) Segment 1

(c) Segment 2

(d) Segment 3

(e) Segment 4

(f) Segment 5

Figure 21: Example segmentation of Censa 214 with HGOA

6.0 DISCUSSION

This chapter provides further qualitative analysis based on the results presented in Chapter 5. A basic conclusion regarding the effects of limited modality data on the DeepMedic models and justification for why the generative model and HGOA algorithms were unsuccessful with the chronic data is presented. We also discuss the performance of each algorithm on chronic and stroke data, then finally suggest two possible avenues for extending this research.

6.1 PERFORMANCE AS INPUT DATA IS REDUCED

When applied to the matched acute training and testing data from ISLES 2015, DeepMedic demonstrated a reduction in performance as input modalities were removed. In [13] DeepMedic was reported to achieve an average Dice score of 0.59 when using all 4 channels: T1, T1c, T2, and FLAIR. When tested here with three channels, removing T1c, it achieves an average Dice score of 0.37. When the FLAIR channel was removed for the T1/T2 model the average Dice score dropped again to 0.27, and either T2 or T1 alone achieved averages of 0.22 and 0.16 respectively. Interestingly, combinations of T1 and FLAIR, T2 and FLAIR, or even FLAIR alone achieved the same average Dice score as the three-channel model, 0.37. This suggests that while performance certainly tends to drop as channels are removed, this algorithm may be overly tuned for T1c and FLAIR data, as any models containing those modalities performed the best, or that T1c and FLAIR imaging are inherently more valuable for viewing stroke lesions.

Results using the generative model to segment the acute ISLES 2015 data were consistent with those of DeepMedic. The algorithm generally lost performance as the number of input channels was reduced, and showed some dependence on the FLAIR images. The top performer was the

three channel combination with an average Dice score of 0.25. All of the two channel combinations performed worse, but with both combinations that included FLAIR images performing better than the T1/T2 combination. T2/FLAIR and T1/FLAIR achieved average Dice scores of 0.21 and 0.14 respectively, T1/T2 performed significantly worse with an average score of only 0.04. Generally, this algorithm performed even worse with only one channel, however the FLAIR only test performed surprisingly well with an average score of 0.19, almost as high as theT2/FLAIR combination. Again, this result suggests that FLAIR images are especially important to the success of the generative model, as they were with DeepMedic. At time of writing no segmentation results of the acute ISLES 2015 data have been published, but in [7] this algorithm was tested using a different 4 channel acute stroke dataset and reported average Dice scores of up to 0.78.

When applied to the chronic dataset, both DeepMedic and the generative model, showed generally poor performance. The low performance in these cases can be partially explained by mismatched disease states. Both DeepMedic and the generative model were designed for acute stroke images, which typically appear with different voxel intensities than chronic stroke damage. Additionally, the DeepMedic models used here were trained with all acute stroke data since not enough chronic data was available for training a model, resulting in models that would be looking for characteristics of acute strokes that may not be generalizable to chronic strokes.

However, DeepMedic still showed a general decrease in performance as the number of input modalities was reduced. The DeepMedic model trained with three modalities achieved an average Dice score of 0.2, and the two worst performers T2 and FLAIR achieved average Dice scores of 0.05 and 0.06 There was one exception, the T1 only model achieved an average Dice score of 0.32, since all DeepMedic models were trained using acute stroke data it is unlikely for these results to be reliable in general.

Using the chronic dataset, the generative model did not show the same decrease as input modalities were removed. These results suggest that the T2 weighted images are the most useful, as T2 and the combination of T1 and T2 were the top two performers. However, all data combinations performed poorly with average Dice scores ranging from 0.1 to 0.18.

6.2 SEGMENTATION FAILURES WITH THE GENERATIVE MODEL

The generative model performed significantly worse than was reported in [7], which ranged from Dice scores of 0.5 to 0.8 when the model was generalized to their stroke dataset. There are several factors that may be related to the decreased performance. Most notably, the only change made to the algorithm in [7] when applying it to the stroke data was to remove the requirement that all lesions in T1c images must be contained within lesions detected in the T2 and FLAIR channels. This suggests that the generative model may be particularly reliant on T1c data, which was not available for these experiments.

Additionally, the code for this algorithm that was made available at [15] was not the complete code tested in [7]. It was missing spatial regularization through Markov Random Fields, and part of the discriminative model extensions, which were the two main contributions of [7]. Markov Random Fields especially were shown to greatly reduce the error rate, and thus increase the Dice score of the final segmentation.

6.3 SEGMENTATION FAILURES WITH DEEPMEDIC

When applied to the acute dataset, DeepMedic exhibited a high variance with each model. Looking at results with the three-channel model there are some patients that DeepMedic achieved a very high Dice score of 0.9, while there are also several that had Dice scores of 0. Some of those low scoring patients had lesions that appeared similar to a chronic stroke lesion such as the image provided as Figure 4. In that patient, the lesion area appears with low intensity values, closer to 0, whereas most of the acute stroke images present with high intensity values closer to 1, and Chronic stroke lesions tend to present low intensity lesions as well. The remaining low scoring patients have very small lesions, suggesting that DeepMedic also has difficulty learning features that successfully detect very small lesions.

6.4 LINDA LIMITATION TO T1 AND CHRONIC IMAGES

Since the publicly available LINDA model was trained with only T1 images containing chronic stroke lesions, it could not be tested with T2, FLAIR, or multi-modal data. Testing it with the T1 data available, LINDA achieved an average Dice score of 0.44. This data was generated at a different lab with a different scanner than the data used to train LINDA and exhibited a Dice reduction greater than the 0.02 reported in [4] where Pustina et al. postulate that this small drop means LINDA is very generalizable to other labs data. However, in this test the greater drop may be better explained by the individual results. Some patients lesions were successfully segmented with Dice scores ranging from 0.8 to 0.9, and there were several that were complete failures with a Dice score of 0. Most of these failed patients had very small lesions, such as the one shown in Figure 19. suggesting that LINDA is simply unable to find these extremely small lesions.

6.5 SEGMENTATION FAILURES WITH HGOA

It is difficult to diagnose why this implementation of HGO was not able to generate any useful segmentations. Since the one used here was written based on the description in [19] as opposed to having the original source code it is possible that some error was made during writing, such as making a mistake in one of the Histogram-based segmentation steps, or making an inaccurate assumption. Likewise, there are several key parameters used in the algorithm that are not described extensively in that document. After each of the first several iterations of the algorithm a random function is added to the image that is reported to greatly increase the likelihood of the optimization algorithm converging, and therefore generating a segmentation. Very little detail about this random function is given, making it difficult to duplicate independantly.

It is also likely that this algorithm is highly specific to DWI images. If the iterative HGOA process converges the output segmentation will contain the number of segments that the objective function was set to look for at the beginning. The algorithm then assigns each segment to either healthy or lesion based on the intensity width. Nabizadeh simply defines the lesion segment width to be 1.8, with little explanation of how that number was reached. It is likely that number is very dependant on the type of modality being used, and would be difficult to pick an appropriate value for other modalities without extensive testing.

6.6 FURTHER RESEARCH

The main avenue for further research is related to DeepMedic's performance on the chronic stroke dataset. Since there were only 24 patients with Chronic strokes available, and only a small subset of those had FLAIR, T1, and T2 data available there was a limited amount of testing possible. With such a small dataset, it is not practical to divide it into training and testing data of any significant size. Ideally, with a chronic stroke dataset at least double in size new DeepMedic models could be trained to test see if its difficulty identifying chronic stroke data is inherent to the algorithm itself, or due to the fact that the models used here were trained with acute stroke images.

Additionally, LINDA was only trained using T1 data, and unable to segment any other modalities. Pustina et al. indicate in [20] that they may make training scripts for new LINDA models available to other research groups in the future. They also indicate that to successfully train a new model at least 30 images must be available, so a large dataset of multi-modal data would also allow for creating new LINDA models using the other image modalities, and possibly a multi-modal LINDA model.

APPENDIX

DEEPMEDIC CONFIGURATION FILES

This Appendix contains copies of the DeepMedic configuration files used to create and train the 3 channel models used in this paper. They can easily be adjusted for a different number of channels by adjusting the variable numberOfInputChannels in the model configuration file, and adding or remocing arguments to the channels variable in the training file. Section A.1 contains a configuration file that controls various model level parameters such as the number of layers and feature maps per layer. Section A.2 contains a the configuration file read by DeepMedic at training time and controls parameters such as the dropout anjd learning rates.

Most of both configuration files are identical to the default ones distributed with the DeepMedic source code at [12], with changes to some values for better segmentation of stroke data as described in [13].

A.1 MODEL CREATION

```
#Default values are set internally, if the corresponding
    parameter is not found in the configuration file.

#[Optional but highly suggested] The name will be used in the
    filenames when saving the model.
```

```
#Default: "cnnModel"
modelName = "deepMedic-FULL"

#[Required] The main folder that the output will be placed.
folderForOutput = "../../../output/"


#================ MODEL PARAMETERS ================

#[Required] The number of classes in the task. Including
    background!
numberOfOutputClasses = 2
#[Required] The number of input channels, eg number of MRI
    modalities.
numberOfInputChannels = 3

#++++++++++++Normal pathway++++++++++++
#[Required] This list should have as many entries as the number
    of layers I want the normal-pathway to have.
#Each entry is an integer that specifies the number of Feature
    Maps to use in each of the layers.
numberFMsPerLayerNormal = [30, 30, 40, 40, 40, 40, 50, 50]
#[Required] This list should have as many entries as the number
    of layers in the normal pathway.
#Each entry should be a sublist with 3 entries. These should
    specify the dimensions of the kernel at the corresponding
    layer.
kernelDimPerLayerNormal = [[3,3,3], [3,3,3], [3,3,3], [3,3,3],
    [3,3,3], [3,3,3], [3,3,3], [3,3,3]]
```

```
#++++++++++Subsampled pathway++++++++++
#[Optional] Specify whether to use a subsampled pathway. If False
    , all subsampled-related parameters will be read but
    disregarded in the model-construction.
#Default: False
useSubsampledPathway = True

#[Optionals] The below parameters specify the subsampled-pathway
    architecture in a similar way as the normal.
#If they are ommitted and useSubsampledPathway is set to True,
    the subsampled pathway will be made similar to the normal
    pathway (suggested for easy use).
#[WARN] Subsampled pathway MUST have the same size of receptive
    field as the normal. Limitation in the code. User could easily
     specify different number of FMs. But care must be given if
    number of layers is changed. In this case, kernel sizes should
     also be adjusted to achieve same size of Rec.Field.
numberFMsPerLayerSubsampled = [30, 30, 40, 40, 40, 40, 50, 50]
kernelDimPerLayerSubsampled = [[3,3,3], [3,3,3], [3,3,3],
    [3,3,3], [3,3,3], [3,3,3], [3,3,3], [3,3,3]]

#[Optional] How much to downsample the image that the subsampled-
    pathway processes.
#Default: [3,3,3]
subsampleFactor = [3,3,3]

#++++++++++Extra FC Layers++++++++++
#[Optional] After the last layers of the normal and subsampled
    pathways are concatenated, additional Fully Connected layers
    can be added before the final classification layer.
```

```
#Specify a list, with as many entries as the number of ADDITIONAL
    FC layers (other than the classification layer) to add. The
    entries specify the number of Feature Maps to use.
#Default: []
numberFMsPerLayerFC = [150, 150]

#++++++++++++Size of Image Segments++++++++++++
#DeepMedic does not process patches of the image, but larger
    image-segments. Specify their size here.

#[Required] Size of training segments influence the captured
    distribution of samples from the different classes (see
    DeepMedic paper)
segmentsDimTrain = [25,25,25]
#[Optional] The size of segments to use during the validation-on-
    samples process that is performed throughout training if
    requested.
#Default: equal to receptive field, to validate on patches.
segmentsDimVal = [17,17,17]
#[Optional] Bigger image segments for Inference are safe to use
    and only speed up the process. Only limitation is the GPU
    memory.
#Default: equal to the training segment.
segmentsDimInference = [45,45,45]

#++++++++++++Batch Sizes++++++++++++
#[Required] The number of segments to create a batch.
#The samples in a training-batch are all processed and one
    optimization step is performed.
#Larger batches approximate the total data better and should
```

```
    positively impact optimization but are computationally more
    expensive (time and memory).
batchSizeTrain = 10
#[Optionals] Batch sizes for validation and inference only
    influence the speed. The bigger the better. Depends on the
    segment size and the model size how big batches can be fit in
    memory.
#Default: Equal to train-batch size.
batchSizeVal = 48
batchSizeInfer = 10

#[Optionals] Dropout Rates on the input connections of the
    various layers. Each list should have as many entries as the
    number of layers in the corresponding pathway.
# 0 = no dropout. 1= 100% drop of the neurons. Empty list for no
    dropout.
#Default: []
dropoutRatesNormal = []
dropoutRatesSubsampled = []
#Default: 50% dropout on every Fully Connected layer except for
    the first one after the concatenation
#Note: The list for FC rates should have one additional entry in
    comparison to "numberFMsPerLayerFC", for the classification
    layer.
dropoutRatesFc = [0.0, 0.5, 0.5] # +1 for the classification
    layer!

#[Optionals] Regularization L1 and L2.
#Defaults: L1_reg = 0.000001, L2_reg = 0.0001
L1_reg = 0.000001
```

```
L2_reg = 0.0001

#[Optional] Initialization method of the kernel weights. Specify
   0 for classic, from the normal distribution N(0, 0.01).
   Otherwise specify 1 for the method of He et al from "Delving
   Deep into Rectifiers".
#Default: 1
initializeClassic0orDelving1 = 1
#[Optional] Activation Function for all convolutional layers.
   Specify 0 for ReLU, 1 for PreLU.
#Default: 1
relu0orPrelu1 = 0


#[Optional] Batch Normalization uses a rolling average of the mus
    and std for inference. Specify over how many batches (
   optimization steps) this rolling average should be taken.
#Default : 60 (in our usual settings, with batchsize=10, segments
    per training subepoch=1000, and subepochs per epoch=20, this
   averages over 5 epochs).
rollAverageForBNOverThatManyBatches = 60

#++++++++++Optimization++++++++++
#[Optionals]
#Initial Learning Rate. Default: 0.001.
learningRate = 0.01
#Optimizer to use. 0 for classic SGD, 1 for Adam, 2 for RmsProp.
   Default: 2
sgd0orAdam1orRms2 = 2
#Type of momentum to use. 0 for standard momentum, 1 for Nesterov
```

```
    . Default: 1
classicMom0OrNesterov1 = 1
#Momentum Value to use. Default: 0.6
momentumValue = 0.6
#Non-Normalized (0) or Normalized momentum (1). Bear in mind that
    Normalized mom may result in smaller gradients and might need
    relatively higher Learning Rate. Default: 1
momNonNorm0orNormalized1 = 1
#Parameters for RmsProp. Default: rho=0.9, e=10**(-4) (1e-6 blew
   up the gradients. Haven't tried 1e-5 yet).
rhoRms = 0.9
epsilonRms = 10**(-4)
```

A.2 TRAINING

```
#Default values are set internally, if the corresponding
   parameter is not found in the configuration file.

#[Optional but highly suggested] The name will be used for saving
    the models,logs and results.
#Default: "trainSession"
sessionName = "trainSessionDeepMedic-FULL"

#[Required] The main folder that the output will be placed.
folderForOutput = "../../../output/"

#[Optional] The path to the saved CNN-model to use for training.
   Optional in the case the the model is specified from command
   line with the -model option. In this case, this entry file of
   the config file will be disregarded, and the one from the
```

```
    command line will be used.
cnnModelFilePath = "../../../output/models/placeholder"


#======================Training
    ====================================

#++++++++++++Input++++++++++++

#[Required] A list that should contain as many entries as the
    channels of the input image (eg multi-modal MRI). The entries
    should be paths to files. Those files should be listing the
    paths to the corresponding channels for each training-case. (
    see example files).
channelsTraining = ["./trainChannels_flair.cfg", "./
    trainChannels_t1c.cfg", "./trainChannels_t2.cfg"]

#[Required] The path to a file which should list paths to the
    Ground Truth labels of each training case.
gtLabelsTraining = "./trainGtLabels.cfg"

#++++++++++++Sampling++++++++++

#[Optional] The path to a file, which should list paths to the
    Region-Of-Interest masks for each training case.
#If ROI masks are provided, under default-sampling settings, the
    training samples will be extracted only within it. Otherwise
    from whole volume.
#This mask is also used for calculating mu and std intensities
    for intensity-augmentation, if performed.
```

```
#roiMasksTraining = "./trainRoiMasks.cfg"

#[Optional] The percentage of training samples to extract from
   foreground (GT-labels).
#Default: 0.5
percentOfSamplesToExtractPositiveTrain = 0.5

#++++++++++Advanced Sampling++++++++++

#[Optional] True in order to use default sampling for training.
   In this case, positive samples are extracted from within the
   GT mask.
#Negative samples are then extracted from the ROI (or full volume
   ), excluding the GT.
#NOTE: Advanced options are disabled if default settings are used
   .
#Default: True
useDefaultTrainingSamplingFromGtAndRoi = True

"""
#++Adv.Sampl. Options. These are disabled if Using-Default-
   Training-Sampling.++
#Weight-maps, that will define areas of the image to sample more.
    Separate for training and negative samples.
#Note: Weight-maps dont have to be normalized.
#Implementation Note: Internally, these are what is actually used
    for sampling. Under default settings, the pos-weight-map is
   passed GT-labels, neg-weight-map passed the ROI.
#weightedMapsForPosSamplingTrain = "./trainGtLabels.cfg"
#weightedMapsForNegSamplingTrain = "./trainRoiMasks.cfg"
```

```
#Specify if the weight-maps are actually weights (True). If False
    , we assume they are GT/ROI. And subtract GT from ROI to do
    the negative-sampling.
samplingMasksAreProbMapsTrain = False
"""

#++++++++++++Training Cycle (see documentation)++++++++++++

#[Optionals but highly suggested as they are model dependent.]
#How many epochs to train for. Default: 35
numberOfEpochs=35
#How many subepochs comprise an epoch. Every subepoch I get
    Accuracy reported. Default: 20
numberOfSubepochs = 20
#Every subepoch, load the images from that many cases and extract
     new training samples. Default: 50
numOfCasesLoadedPerSubepoch = 50
#Every subepoch, extract in total this many segments and load
    them on the GPU. Memory Limitated. Default: 1000
#Note: This number in combination with the batchSizeTraining,
    define the number of optimization steps per subepoch (=
    NumOfSegmentsOnGpu / BatchSize).
numberTrainingSegmentsLoadedOnGpuPerSubep = 1000

#++++++++++++Learning Rate Schedule++++++++++++

#[Optional] The type of schedule to use for Learning Rate
    annealing.
#0=Stable Decrease. 1=Auto (Lower LR when validation accuracy
    plateaus. Requires validation-on-samples). 2=Lower at
```

```
    predefined epochs. 3=Exponentially decrease LR, linearly
    increase Mom.
#NOTE: Training Schedule is very important. We suggest running
    stable and observing training error, then lower LR when it
    plateaus. Otherwise, use exponential but make sure to train
    for enough epochs.
stable0orAuto1orPredefined2orExponential3LrSchedule = 2

#[For Stable + Auto + Predefined] By how much to divide LR when
    lowering. Default: 2
whenDecreasingDivideLrBy = 2.0

#[For Stable + Auto] How many epochs to wait before decreasing
    again. Set Zero to never lower LR. Default: 3
#numEpochsToWaitBeforeLoweringLr = 3
#[For Auto] If validation accuracy increases more than this much,
     reset the waiting counter. Default: 0.0005
#minIncreaseInValidationAccuracyThatResetsWaiting = 0.0005

#[Required for Predefined] At which epochs to lower LR.
predefinedSchedule = [12, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43,
    46]

#[Required for Exponential] [First epoch to start lowering from,
    value for LR to reach at last epoch, value for Mom to reach at
     last epoch]
exponentialSchedForLrAndMom = [12, 1.0/(2**(7)), 0.9]

#++++++++++Data Augmentation++++++++++
```

```
#[Optional] Specify whether to reflect the images by 50%
    probability in respect to the X/Y/Z axis. Default: [False,
    False, False]
reflectImagesPerAxis = [True,False,False]

#[Optional] Augmentation by changing the mean and std of training
     samples. Default: False
performIntAugm = False
#I' = (I + shift) * multi
#[Optionals] We sample the "shift" and "multi" variable for each
    sample from a Gaussian distribution. Specify the mu and std.
#Defaults : [0, 0.1] and [1.,0.]
sampleIntAugmShiftWithMuAndStd = [0, 0.1]
sampleIntAugmMultiWithMuAndStd = [1., 0.0]


#==================Validation====================

#[Optionals] Specify whether to perform validation on samples and
     full-inference every few epochs. Default: False for both.
performValidationOnSamplesThroughoutTraining = False
performFullInferenceOnValidationImagesEveryFewEpochs = True

#[Required] Similar to corresponding parameter for training, but
    points to cases for validation.
channelsValidation = ["./validation/validationChannels_flair.cfg"
    , "./validation/validationChannels_t1c.cfg", "./validation/
    validationChannels_t2.cfg"]

#[Required for validation on samples, optional for full-inference
```

```
    ] Similar to corresponding parameter for training, but points
    to cases for validation.
gtLabelsValidation = "./validation/validationGtLabels.cfg"

#[Required] Similar to corresponding parameter for training. Only
     influences how accurately the validation samples will
    represent whole data. Memory bounded.
#Default: 3000
numberValidationSegmentsLoadedOnGpuPerSubep = 5000

#[Optional] Similar to corresponding parameter for training
#roiMasksValidation = "./validation/validationRoiMasks.cfg"

#+++++Advanced Validation Sampling+++++:

#[Optional] True in order to use default sampling for validation.
     Default is uniform sampling from ROI.
#NOTE: Advanced options are disabled if default settings are used
    .
#Default: True
useDefaultUniformValidationSampling = True

"""
#++Adv.Sampl. Options. These are disabled if Using-Default-
    Validation-Sampling.++
#Similar parameters to the training case.
percentOfSamplesToExtractPositiveVal = 0.0
weightedMapsForPosSamplingVal = None
weightedMapsForNegSamplingVal = "./validation/validationRoiMasks.
    cfg"
```

```
samplingMasksAreProbMapsVal = False
"""

#+++++Full-Inference on validation cases+++++
#[Optional] How often (epochs) to perform full inference. It is
    time consuming... Default: 1
numberOfEpochsBetweenFullInferenceOnValImages = 5

#[Optionals] Specify whether to save the segmentation and
    probability maps for each class. Default: True to all
saveSegmentationVal = True
saveProbMapsForEachClassVal = [True, True, True, True, True]

#[Required if requested to save results] The path to a file,
    which should list names for each validation case, to name the
    results after.
namesForPredictionsPerCaseVal = "./validation/
    validationNamesOfPredictions.cfg"

#--Feature Maps--
#Feature maps can also be saved, but section is omitted here. See
     testing configuration.

#==========Generic============
#[Optional] Pad images to fully convolve. Default: True
padInputImagesBool = True
```

BIBLIOGRAPHY

[1] K. Kamnitsas, C. Ledig, V. F. Newcombe, J. P. Simpson, A. D. Kane, D. K. Menon, D. Rueckert, and B. Glocker, "Efficient Multi-Scale 3D CNN With Fully Connected CRF for Accurate Brain Lesion Segmentation," 2016.

[2] B. H. Menze, K. Van Leemput, D. Lashkari, M.-A. Weber, N. Ayache, and P. Golland, "A Generative Model for Brain Tumor Segmentation in Multi-Modal Images," *Med Image Comput Comput Assist Interv.*, pp. 151–159, 2010.

[3] N. Nabizadeh, J. Nigel, and C. Wright, "Histogram-Based Gravitational Optimization Algorithm on Single MR Modality for Automatic Brain Lesion Detection and Segmentation," *Expert Systems With Applications*, pp. 7820–7836, 2014.

[4] D. Pustina, H. B. Coslett, P. E. Turkeltaub, N. Tustison, M. F. Schwartz, and B. Avants, "Automated Segmentation of Chronic Stroke Lesions Using LINDA: Lesion Identification With Neighborhood Data Analysis," *Human Brain Mapping*, pp. 1405–1421, 2016.

[5] J. Liu, M. Li, J. Wang, F. Wu, T. Liu, and Y. Pan, "A Survey of MRI-Based Brain Tumor Segmentation Methods," *Tsinghua Science and Technology*, pp. 578–595, 2014.

[6] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotbloom, and R. Wiest, "The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)," *IEEE Transactions on Medical Imaging*, p. 33, 2014. <hal-00935640v1>.

[7] B. H. Menze, K. Van Leemput, D. Lashkari, T. Riklin-Raviv, E. Geremia, E. Alberts, P. Gurber, S. Wegener, M. Weber, G. Székely, N. Ayache, and P. Golland, "A Generative Probabilistic Model and Discriminative Extensions for Brain Lesion Segmentation - With Application to Tumor and Stroke," *IEEE Transactions on Medical Imaging*, pp. 933–946, 2016.

[8] M. L. Seghier, A. Ramlackhansingh, J. Crinion, A. P. Leff, and C. J. Price, "Lesion Identification Using Unified Segmentation-Normalization Models and Fuzzy Clustering," *NeuroImage*, pp. 1208–1212, 2008.

[9] A. A. Taha and A. Hanbury, "Metrics for Evaluating 3D Medical Image Segmentation: Analysis, Selection, and Tool," *BMC Medical Inaging*, vol. 15, no. 29, 2015.

[10] K. W. Lai, D. Ekashanti, and O. Dewi, eds., *Medical Imaging Technology, Reviews and Computational Applications*. Springer, 2015.

[11] P. Krähenbuühl and V. Koltun, "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials," 2012. arXiv preprint arXiv:1210.5644.

[12] K. Kamnitsas, "The DeepMedic," June 2016. Retrieved from Github: https://github.com/Kamnitsask/deepmedic.

[13] K. Kamnitsas, L. Chen, C. Ledig, D. Rueckert, and B. Glocker, "Multi-Scale 3D Convolutional Neural Netowrks for Lesion Segmentation in Brain MRI," 2016. Retrieved from Imperial College Website: http://www.doc.ic.ac.uk/ bglocker/pdfs/kamnitsas2015isles.pdf.

[14] X. Li, "DICOM to NifTI Converter, NifTI Tool and Viewer," 2013. Retrieved from MathWorks File Exchange: https://www.mathworks.com/matlabcentral/fileexchange/42997-dicom-to-nifti-converter–nifti-tool-and-viewer.

[15] E. Alberts, "Expectation-Maximisation Algorithm For Brain Tumor Segmentation," June 2016. Retrieved from Bitbucket: https://bitbucket.org/s0216660/brain_tumor_segmentation_em/overview.

[16] S. Klein and M. Staring, "Elastix," May 2016. Retrieved from Image Sciences Institute Website.

[17] D. Shamonin, E. Bron, B. Lelieveldt, M. Smits, S. Klein, and M. Staring, "Fast Parallel Image Registration on CPU and GPU for Diagnostic Classification of Alzheimers Disease," vol. 7, no. 50, pp. 1–15, 2014.

[18] S. Klein, M. Staring, K. Murphy, M. Viergever, and J. Pluim, " Elastix: A Toolbox for Intensity Based Medical Image Registration," *IEEE Transactions on Medical Imaging*, vol. 29, no. 1, pp. 196–205, 2010.

[19] N. Nabizadeh, *Automated Brain Lesion Detection and Segmentation Using Magnetic Resonance Images*. PhD thesis, University of Miami, 2015. Open Access Dissertations paper 1409.

[20] D. Pustina, H. Coslett, P. Turkeltaub, N. Tustison, M. Schwartz, and B. Avants, "Lesion Identification With Neighborhood Data Analysis," 2016. Retrieved from Github: https://github.com/dorianps/LINDA.

[21] P. A. Yushkevich, J. Piven, H. C. Hazlett, R. G. Smith, S. Ho, J. C. Gee, and G. Gerig, "User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability," vol. 31, no. 3, pp. 1116–1128, 2006. www.itksnap.org.

[22] J. Shen, "Tools for NifTI and ANALYZE Images," 2014. Retrieved from MathWorks File Exchange: https://www.mathworks.com/matlabcentral/fileexchange/8797-tools-for-nifti-and-analyze-image.